ISSN 0265-2919
90p 81

THE HOME COMPUTER ADVANCED COURSE

MAKING THE MOST OF YOUR MICRO

An ORBIS Publication
IR£ 1.15 Aus $2.15 NZ $2.65 SA R2.45 Sing $4.50

CONTENTS

Next Week

- **The Workshop MIDI interface we designed earlier in the course was based on a 6502 processor. We begin a two-part look at how to modify this design for interfacing with the Amstrad CPC464.**
- **In our series on gambling applications, we discuss the many ways that computers are used by betting organisations.**
- **The Amstrad CPC 6128 is the latest computer from a company that is steadily building up its stature in the home micro market.**

QUIZ

1) Which key do you press to initiate an audio playback on The Music System?
2) Which operator has the greatest precedent in our Spreadsheet program?
3) What package was written by Professor Frank George to assist punters?
4) A major problem with ink jet printers has recently been solved. What was it?

Answers To Last Week's Quiz
1) The AND operation.
2) In the Amstrad operating system, a 'kick' is an increment of the event count.
3) The 'cropping tool' icon allows you to shrink, enlarge or trim graphics to fit the available space.

Coming Up

- A detailed look at the MS-DOS operating system.
- A series of articles investigating the wide range of careers involving computers.

Editor Stephen Cooke; **Art Editor** Claudia Zeff; **Deputy Editor** Steve Colwill; **Production Editor** Bobby Pickering; **Designer** Julian Dorr; **Staff Writer** Steve Malone; **Art Assistant** Caroline Clayton; **Sub Editor** Jon Kaye; **Contributors** Chris Honey, Chris Laing, Dougie Bern, Tony Drapkin, Pete Connor, Mike Curtis, Steve Cooke, Steve Colwill, Steve Malone, Steven Vickers, David Mudd; **Software Consultants** Pilot Software City; **Group Art Director** Perry Neville; **Managing Director** Stephen England; **Published by Orbis Publishing Ltd: Editorial Director** Brian Innes; **Project Development** Peter Brookesmith; **Executive Editor** Maurice Geller; **Production Assistant** Susan Brown; **Subscription Manager** Christine Allen; **Designed and produced by Bunch Partworks Ltd; Editorial Office** 14 Rathbone Place, London W1P 1DE; **© APSIF Copenhagen 1985; © Orbis Publishing Ltd 1985: Typeset by Universe; Reproduction by Mullis Morgan Ltd; Printed in Great Britain by Heanor Gate Printing Ltd, Derby**

HOW TO OBTAIN ISSUES AND BINDERS FOR THE HOME COMPUTER ADVANCED COURSE – Issues can be obtained by placing an order with your newsagent or direct from our subscription department. If you have any difficulty obtaining any back issues from your newsagent, please write to us stating the issue(s) required and enclosing a cheque for the cover price of the issue(s). **AUSTRALIA** – please write to: Gordon & Gotch (Aus) Ltd, 114 William Street, PO Box 767G, Melbourne, Victoria 3001. **MALTA, NEW ZEALAND & SOUTH AFRICA** – Back numbers are available at cover price from your newsagent. In case of difficulty, write to the address given for binders.
UK/EIRE – Issue Price: 90p/IR£1.15. Subscription: 6 months: £26.00. 1 Year: £52.00. Binder: please send £3.95 per binder, or take advantage of our special offer in early issues. **EUROPE** – Issue Price: 90p. Subscription: 6 months air: £44.72. Surface: £36.14. 1 year air: £89.44. Surface: £72.28. Binder: £5.00. Airmail: £8.25. **MALTA** – Obtain binders from your newsagent or Miller (Malta) Ltd, MA Vassalli Street, Valetta, Malta. Price: £3.95. **MIDDLE EAST** – Issue Price: 90p. Subscription: 6 months air: £50.18. Surface: £36.14. 1 year air: £100.36. Surface: £72.28. Binder: £5.00. Airmail: £8.25. **AMERICAS/ASIA/AFRICA** – Issue Price: US/CAN$1.95/90p. Subscription: 6 months air: £59.54. Surface: £36.14. 1 year air: £119.08. Surface: £72.28. Binder: £5.00. Airmail: £9.50. **SOUTH AFRICA** – Issue Price: SA R2.45. Obtain binders from any branch of Central News Agency or Intermag, PO Box 57394, Springfield 2137. **SINGAPORE** – Issue Price: Sing $4.50. Obtain binders from MPH Distributors, 601 Sims Drive, 03-07-21, Singapore 1438. **AUSTRALASIA/FAR EAST** – Issue Price: 90p. Subscription: 6 months air: £64.22. Surface: £36.14. 1 year air £128.44. Surface: £72.28. Binder: £5.00. Airmail: £9.75. **AUSTRALIA** – Issue Price: Aus$2.15. Obtain binders from First Post Pty Ltd, 23 Chandos Street, St Leonards, NSW 2065. **NEW ZEALAND** – Issue Price: NZ$2.65. Obtain binders from your newsagent or Gordon & Gotch (NZ) Ltd, PO Box 1595, Wellington.
ADDRESS FOR BINDERS AND BACK ISSUES – Orbis Publishing Limited, Orbis House, Bedfordbury, London WC2 4BT. Telephone 01-379 5211. Cheques/postal orders should be made payable to Orbis Publishing Limited. Binder prices include postage and packing and prices are in stirling. Back issues are sold at the cover price, and we do not charge carriage in the UK.
NOTE – Binders and back issues are obtainable subject to availability of stocks. Whilst every attempt is made to keep the price of the issues and binders constant, the publishers reserve the right to increase the stated prices at any time when circumstances dictate. Binders depicted in this publication are those produced for the UK and Australian markets only. Binders and Issues may be subject to import duty and/or local taxes, which are not included in the above prices unless stated.
ADDRESS FOR SUBSCRIPTIONS – Orbis Publishing Limited, Hurst Farm, Baydon Road, Lambourn Woodlands, Newbury Berks, RG16 7TW. Telephone: 0488-72666. All cheques/postal orders should be made payable to Orbis Publishing Limited. Postage and packaging is included in subscription rates, and prices are given in sterling.

COVER PHOTOGRAPHY BY CRISPIN THOMAS

COURSE WORK

Gambling in general, and on horse racing in particular, is a big business that's spawned much literature advising punters on how to make the most of their bets. We begin here a four-part series on the application of microcomputers to gambling, starting with a look at four software packages for the horse racing devotee.

Sporting Chance SPORTING PICTURES LTD.
The great number of different factors affecting the outcome of a race such as this one tend to encourage an undisciplined and unprofitable approach to betting. However, the computer can act as a useful tool for the serious punter, reducing outlay and maximising potential profit by spotting combinations of good form and attractive odds

Considering the immense popularity of gambling, it's scarcely surprising to learn that the introduction of computers has spurred many programmers into devising algorithms aimed at maximising profits, minimising losses, and predicting the outcome of the 2.30 at Newmarket. In this four-part series, we'll examine some of the techniques and theory behind computer gambling applications, and do a little forecasting of our own on the subject of whether or not your micro will make you a millionaire.

In this instalment, we examine four different software packages, each of which claims to be of use to the punter. We also take a look at one of the principal sources of data for the gambler concerned with using such programs — *The Sporting Life* — and assess the likely profit (or loss!) margins for each title. In succeeding instalments, we'll look at the use of computers at the racecourse, the science of statistics and probability theory, and relevant artificial intelligence applications.

COURSEWINNER

One of the more interesting programs available to the punter is Coursewinner, from Selec Software. It assumes that you have a copy of *The Sporting Life* or a similar specialised racing paper with the relevant data. The factors chosen for analysis are sensible and likely to give an accurate assessment of known form. A pleasing feature here is the ability to enter up to three different 'speed ratings' as published in the sporting and popular press or as available from ratings specialists such as Timeform.

The menu-driven program is easy to use although error-trapping is restricted. A thoughtful touch is the provision of a warm start without loss of current data so that accidentally hitting Escape (BBC version) isn't disastrous.

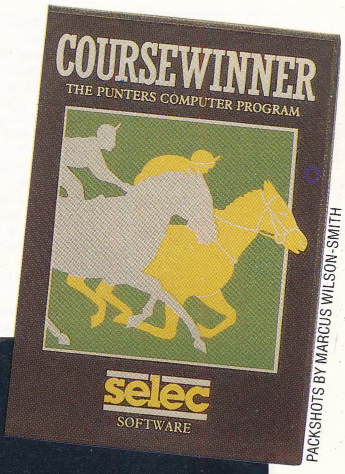
Once the relevant form factors have been entered, you are expected to provide the actual prices on offer, because the program, apart from calculating the horse's 'true odds', attempts to advise as to whether the bookmakers' odds represent good value, in which case it recommends a three-star bet. However, since the betting market is only formed about ten minutes before the start of each race, this is not always possible. Instead, you could enter the betting forecast from the morning papers, thus invalidating the recommended three-star bets, but not necessarily the ratings.

A further option involves adjusting the weighting given to the various form factors in calculating a horse's chances. This too is a useful feature, particularly for the experienced punter.

The package was tested over 'Glorious Goodwood' in 18 races with ten runners or less. The three-star value bets were unspectacular with three winners from 12 selections for a level stake profit after tax of £1.60 to a £1 stake. However, horses given an even money or better chance did well, with three winners out of five for a level stake profit of £3.73, while those rated as 6-4 shots (disconcertingly referred to by the program as 3-2) produced two out of six for a profit of 70p.

Coursewinner is an interesting, well-designed program, based on sound principles which, on the basis of an admittedly insignificant statistical sample, seems to offer the thoughtful punter a chance to bet selectively and profitably. Obviously, anyone considering using this or any other system seriously would be well-advised to give it an extensive dry run before investing real money.

Coursewinner: For the Spectrum, Commodore 64, Amstrad CPC micros, Atari, Apple II and BBC Model B Micro
Price: £15.00
Distributors: Selec Software, 37 Councillor Lane, Cheadle, Cheshire



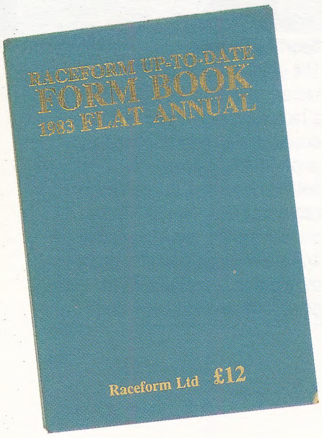
PACKSHOTS BY MARCUS WILSON-SMITH

Using Coursewinner

On loading Coursewinner, you are confronted by a menu; type C and a list of courses appears. Type in the number corresponding to the course at which the race for analysis is to be run, and then enter the distance to the nearest furlong and the official going (to be found at the top of the Racecard). The program now returns to the main menu. Type I to input race data. The program branches to a sub-menu as illustrated with four routines for entering the different kinds of data required. Work through each routine, simply entering data in the form requested by the program. In the section headed 'speed factors', three of the following rating systems should be used: Stopwatch (Sporting Life), Form Ratings (Sporting Life), Spotform (Daily Mirror), Formcast (Daily Mail) and Timeform (by subscription). Of these, Timeform ratings are recommended.

On returning to the main menu, type P. A request for starting prices will appear. If a real betting show is not available then use the betting forecast; but remember that this will devalue the recommended three-star bets. It will not necessarily invalidate the program's calculation of the 'real odds' of the runners.

When the main menu reappears, type for an analysis of the race, as illustrated. The lower the calculated odds, the better the chance of the horse concerned. Initially, it is probably worth restricting bets to horses with calculated odds of 1-1 (even money) or better, since these should win 50 per cent of their races with a few long losing runs. If the bookmakers are offering odds of 6-4 or more about them, so much the better. Such bets represent real value for money, the principle that lies at the heart of all successful punting



SPRINT FORMULA

Brimardon Computer Racing Service provides a number of services for the thoughtful punter who wishes to get more out of his micro. These include guides to writing your own programs, ready written programs that are adaptable to individual requirements, programs tailor-made to clients' specifications and Sprint Formula. The latter is a package aimed at the experienced punter who knows the value in sprints of speed figures based on race times. It focuses on all-aged handicaps up to seven furlongs because these notoriously difficult races offer open betting. The authors argue, with some justification, that this provides a golden opportunity for the punter who has a reliable means of assessing merit.

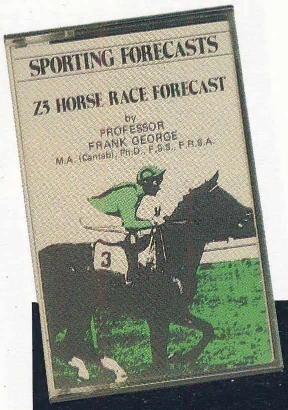
The horse's speed figure, which is central to the formula, may be enhanced or devalued by factors for the going (i.e. the state of the course), distance and class of race in which the figure was obtained.

Sprint Formula: For the BBC Model B, Acorn Electron, Spectrum, Commodore 64, Amstrad CPC range, Tandy, Dragon, Oric and Atari.
Price:£25.00
Distributors: Brimardon Computer Racing Service, 48 Pierremont Road, Darlington

In addition, recent form figures are assessed and a weighting given against such obvious pointers as a second place last time out. The authors consider that these runners often start at cramped odds which don't represent value for money. The aim is to get winners at longer odds.

The system certainly does this for those prepared to back the two top-rated horses. Brimardon supplies details of the season's results that can be checked. So far, the top two have produced 28 winners form 123 bets for a level stake profit after tax of 79 points, representing 62 per cent on turnover. The average SP of winners was just over 13-2. Those prepared to endure the occasional losing run were rewarded with winners at up to 20-1.

The authors admit that, because of memory limitations, the output from Sprint Formula is not pretty, while the input routines are a trifle rough at the edges. Error-trapping is non-existent and it is up to you to avoid entering garbage. To compensate for this, the program comes with extensive, helpful documentations. Altogether, it is a useful package for the selective punter who will appreciate its cleverness, and it would seem, its successful approach to the problem of real value for money betting.



Z5 HORSE RACE FORECAST

Z5 Horse Race Forecast, by Professor Frank George, also takes its data from *The Sporting Life* and uses speed figures from the *Raceform Handicap Book.* It uses rather less data than Coursewinner 3 and accepts two ratings — the appropriate speed figure plus the form ratings service on page two of *The Sporting Life.*

Information is input in response to a series of prompts. The more obvious errors are trapped and each data item is queried for correctness before entering. The program then rates each horse according to the following scale: excellent bet, very good bet, possible bet, poor bet, outsider bet and eliminate. Unfortunately, no attempt is made to quantify these ratings, so the onus is on the user to test the program stringently in order to establish their true statistical merit.

The disconcerting feature of this program is the frequency with which up to five runners in the same race are given the top ratings in fields of ten runners or less. Here, the author makes two recommendations — either do not bet, or back all the top-rated horses (if it is possible to do so and still show a profit). However, on this basis, the number of real winning opportunities is limited.

Z5 Horse Race Forecast: For the Spectrum, BBC Model B Micro and Commodore 64
Price: £15.00 plus VAT
Distributors: Bureau of Information Science, Chalfont, St Giles

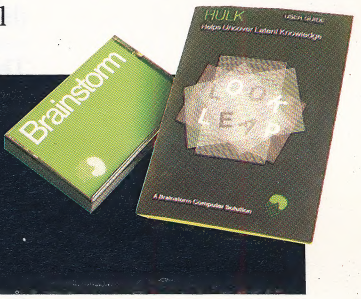
HULK I AND II

Hulk (from Warm Boot Ltd) is not specifically designed to help the computer punter. It's described by its creator as a 'poor man's knowledge engineer', in that it helps you to build your own expert system. If you accept that the successful punter is an expert in the field, why not attempt to embody that expertise in the computer? Hulk is therefore particularly useful to the experienced punter who likes to work to a 'system' — a set of rules based on analysis of past data which, when rigorously applied, produces winners in sufficient numbers to turn in a percentage profit.

The first and most time-consuming task is that of setting up a database. Memory limitations in the BBC Micro version make demands on the ingenuity of the user who must have a clear idea in advance of the factors most likely to be relevant, and then select a specific type of race for analysis. A typical strategy might involve the study of handicaps over a given distance in races of ten runners or less, concentrating on the first three or four in the betting forecast. It is essential to ensure that a statistically representative sample has been chosen.

The data is then analysed by a program called Look, which enables you to develop a set of rules (typically seven or eight) that predict a given hypothesis (in this case whether or not a horse is a winner). Once a satisfactory set of rules has been obtained, these can be applied to furture events by another program, called Leap, which makes its predictions in terms of percentage probability..

Hulk is a fascinating program for the 'system' punter devoted to statistical analysis as a means of finding winners. Using the package, it is possible to devise rule sets that predict winners on a highly selective basis, but with a degree of accuracy well in excess of 60 per cent.



Hulk I and II: Hulk I for the BBC Model B Micro; Hulk II for dBase II, MS-DOS and PC-DOS machines
Price: Hulk I, £25.00; Hulk II, £195.00 (inc VAT)
Distributors: Warm Boot Ltd, Finsbury Business Centre, 40 Bowling Green Lane, London EC1R 0NE, and by Brainstorm Computer Solutions, 103a Seven Sisters Road, London, N7 7QN

Reading The Form

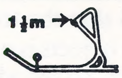
The Racecard gives the time, distance, conditions and value of the race followed by a list of runners. Reading from left to right, the details for each runner are as follows: number, placing in last six races, name, owner, trainer, age, weight carried, jockey and draw. Supplementary information on penalties, course and distance winners, and apprentices' allowances are also to be found here on occasion.

The list of runners is followed by 'stopwatch ratings'. The best time recorded by a horse is expressed in simple numerical form after various calculations have been made to allow for going, weight and so on. A good speed figure is important, since there's no point in backing slow horses. Following this is the betting forecast, which is the experts' ideas of how the punters are likely to bet. It is a rough guide and does not necessarily represent what actually happens when a market is formed.

The Sporting Life Form on the right gives the horse's general profile, including its season's placings, name, age, weight carried, colour, sex, breeding, list of races won (including distance, going and course), total winnings and season's winnings. This is followed by brief notes describing the horse's last three runs

The Racecard

Fourth Race



One mile and a half, for three yrs old and upwards

4.10 THE ALYCIDON STAKES (Listed Race)

£12000 added to stakes
Distributed in accordance with Rule 194 (iii)(a)
(Includes a fourth prize)
for three yrs old and upwards which, at starting, have not won a Pattern race since two yrs old
ONE MILE AND A HALF
£24 to enter, £96 extra if declared to run
Weights: 3-y-o colts and geldings....8st 2lb; fillies 7st 13lb
4-y-o and up colts and geldings.....9st 1lb; fillies 8st 12lb
Penalties, since 2-y-o, a winner of a race value £4000 .... 3lb
Of a race value £8000 .... 6lb
Allowances: 4-y-o and up which, at starting, have not won a race since 1983 and 3-y-o maidens at starting 6lb
' A trophy value £500, at the option of the winner, is included in the value of this race

A **SS**

48 entries, 31 at £24 and 17 at £120.-Closed July 17th, 1985

Owners Prize Money. Winner £7297; Second £2393; Third £1136 ; Fourth £508 (Penalty Value £9489.60)

Form			Trainer	Age	st	lb	Draw
	2	**PARLIAMENT**		**5**	**9**	**7**	**(4)**
		Ch h Lord Gayle (USA) - Harbrook					
11—2231		**Mrs Pamela Stokes**	**(C. R. Nelson, Upper Lambourn)**				P. Cook
D		ROYAL BLUE and WHITE stripes, RED sleeves.					

(Breeder - Shanbally House Stud.)

Very consistent sort. Victory in listed race last time (Ayr 10f) and close up 2nd in Group 3 Westbury Stakes, (Sandown 10f) behind Elegant Air illustrates that he is not without claims.

Form			Trainer	Age	st	lb	Draw
	4	**SHERNAZAR**		**4**	**9**	**1**	**(2)**
		B c Busted - Sharmeen (FR)					
211245—		**H.H. Aga Khan**	**(M. R. Stoute, Newmarket)**				
		GREEN, RED epaulets.					

(Breeder - H.H. Aga Khan.)

Had some very smart form last year, chasing home Commanche Run in the Gordon Stakes here 12f and

SPORTING LIFE FORM

KEY: a—slower than average; b—faster than average; eq—equals average; bl—blinkers; d—disqualified; ow—overweight; £—value to the winner; asterisk denotes apprentice claim, figures appearing after the jockey's name before bracket denote draw position.

2.30—NEWHOLME STAKES (2-y-o). 6f. (£1,244).

AVENTINO (8-11) ch c Cure The Blues — Sovereign Dona by Sovereign Path.

00 BROKERS DREAM (April 7, 5,400gns) (8-11) ch c Free State — Pamora by Blast.

April 27, Sandown, 5f (2-y-o) mdn, good, £2,691: 1 Teetoy(USA) (9-0, 8); 2 Cliveden(USA) (9-0, 11); 3 Sit This One Out (9-0, 10); 10 BROKERS DREAM (9-0, B Thomson, 7), in touch to half-way (33 to 1). 12 Ran. 3l, 1½l, hd, ¾l, 1l. 1m 3.19s (a 2.69s).

April 16, Newmarket, 5f (2-y-o), good, £2,532: 1 Andartis (9-0 , 11); 2 Meadow Moor (9-0 , 8); 3 For Dear Life (9-0 , 5); 9 BROKERS DREAM (9-0 , B Thomson , 7), never beyond mid-division (14 to 1 tchd 20 to 1). 11 Ran. 1½l, nk, ¾l, nk, 2½l. 1m 2.23s (a 2.43s).

3 CONQUERING HERO(USA) (May 2) (8-11) b c Storm Bird — Wave In Glory by Hoist The Flag.

Aug 3, Windsor. See ELNAWAAGI(USA).

311 ELNAWAAGI(USA) (Feb 4, $ 4,000,000) (9-6) b c Roberto(USA) — Gurkhas Band (USA) by Lurullah. **1985, 6f good (Windsor), 6f good to firm (Thirsk). £2,612.**

Aug 3, Windsor, 6f (2-y-o), good, £941: 1 ELNAWAAGI(USA) (9-4 , A Murray , 12), made virtually all, stayed on well when shaken up final furlong (5 to 4 fav op 5 to

41 FEISTY (March 13, 44,000gns) (9-3) b c Pitskelly — Golden Empress by Cavo Doro. **1985, 6f good (Yarmouth). £1,066.**

Aug 8, Yarmouth, 6f (2-y-o) mdn, good, £1,066: 1 FEISTY (9-0 , T Ives , 1), made virtually all, drew clear well over one out, comfortably (100 to 30 op 5 to 2 tchd 7 to 2); 2 Below Zero (9-0 , 5); 3 Pulham Mills (9-0 , 6). 7 Ran. 6l, ½l, ½l, 7l, ½l, 1½l. 1m 16.8s (a 4.2s).

June 17, Windsor, 5f (2-y-o), good to firm, £965: 1 Roaring Riva (8-11 , 14); 2 Dee-Jay-Ess (9-3 , 1); 3 Sitzcarraldo (8-4 , 7*, bl, 8); 4 FEISTY (8-11 , T Ives , 9), always chasing leaders, stayed on inside last (11 to 1 op 5 to 1 tchd 12 to 1); 0 PORTHMINSTER (8-11 , J Reid , bl, 11), (50 to 1). 17 Ran. Sht hd, 2l, 2½l, 2½l, ¾l. 59.9s (a 0.6s).

00 FIRST BILL (Feb 28) (8-11) ch c Nicholas Bill — Angelica by Hornbeam.

Aug 2, Goodwood, 7f (2-y-o) mdn, good, £4,675: 1 New Trojan (9-0 , 11); 2 Mashkour(USA) (9-0 , 8); 3 Bronze Opal(USA) (9-0 , 13); 8 FIRST BILL (9-0 , J Matthias , 9), well placed for over 4f, weakened (50 to 1). 15 Ran. ¾l, 3l, 3l, 1½l, ½l. 1m 31.59s (a 4.99s).

July 13, Salisbury, 7f (2-y-o) mdn, good to firm, £1,697: 1 Atig(FR) (9-0, 17); 2 Ininsky (9-0, 1); 3 Sohail(USA) (9-0, 3); 0 FIRST BILL (8-7 , G Landau , 7*, 12), speed for 4f (33 to 1 op 10 to 1). 19 Ran.

COURTESY OF THE SPORTING LIFE

GOING BY DEFAULT

One of the major criteria of a good adventure game is the apparently independent actions of the characters. Before implementing a tree for this purpose, however, we take a comprehensive look at the object-handling routines we've already discussed.

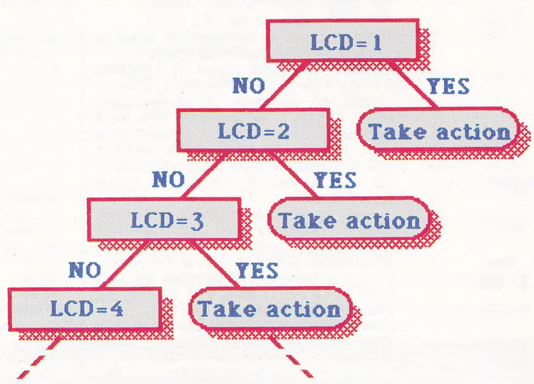
The lines that we added to our basic module in the last instalment (see page 1596) enable the characters to pick up and, to a limited extent, use the objects around them. Before we proceed with designing the remaining decision trees to control character interaction and plot, it's worth looking at the object manipulation process in greater detail.

It's important to realise that if you intend to use these ideas in an adventure game of our own design, you should plan your tree so that messages are not printed to the screen too often. Although lots of messages might seem like a good idea, they can disrupt your progress through the game if they occur too frequently.

As a general rule, you can divide character actions into three levels. The top level involves actions such as giving or taking an object and requires that a message be printed, if you're present, informing you what has happened. The second level involves those actions that don't necessarily affect the plot of the game, but which may nevertheless have messages displayed to help create atmosphere. A message might also be displayed if you enter Examine a character or words to that effect. The third level — altering the value of the 'mood' flag in our own program, for example — never results in a message being printed.

Branch Limitations

Using a tree structure where each node can only branch in two directions can pose serious limitations when testing conditions for a number of different values



It's worth bearing this idea of 'levels' in mind when programming your own game — you can assign a value to each action that determines its level, and then have just one subroutine to deal with printing messages. Such a routine would first check to see if you were present, check the level of the action if you were, and decide whether or not to print a message.

Going back to our object manipulation routine, first try removing line 5010. This will have the effect of ensuring that messages for all characters are printed, regardless of whether they are in the same room as the player or not. You will see that, for the most part, characters concentrate on getting and drinking their 'own' objects, which is what we want if we are to maintain a degree of realism. You can, however, experiment with this by altering the default values in the data statements in lines 6030 and 6040. For example, if you alter the inventories of Fiona Frappe, Steve Swigg, and Molly Mixer to 9, 12, and 8 respectively, and then RUN the program, you will see that the actions in the lounge take a rather different course. Try 'tuning' other characters in this fashion and see what happens.

As well as changing line 5010 as indicated above, you will also find 'tuning' your characters easier by skipping lines 560 to 580 so that each character is processed by each call to the handler routine, rather than left waiting until their flags have been decremented to zero. To do this, simply edit line 550 to read:

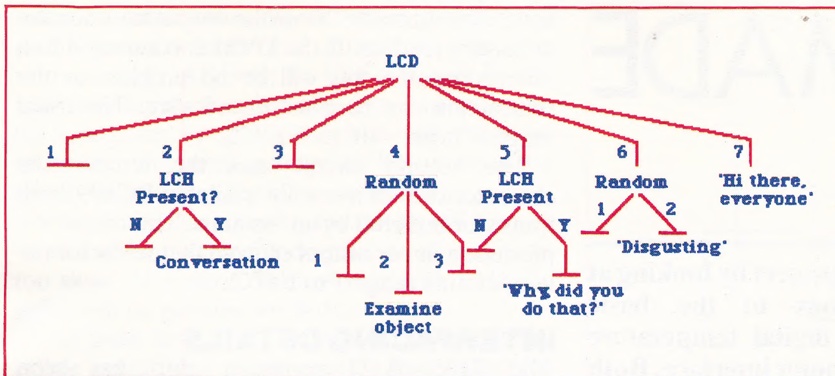
```
550 FOR c=1 TO 6: GOTO 590
```

You will now see the actions of each character displayed one after the other. Remember, if you decide the characters are not being updated quickly enough by the original program, you can always alter the values for the 'handle' flags in lines 6030 and 6040. (If you've forgotten exactly which element of the c array refers to which factor, see page 1492.)

Let's now move on to interaction. We've already DIMensioned the t array in line 190 to hold the data for three trees, each with up to 25 choice nodes. We may find that we need to alter this value, but more importantly, we may also find that the tree structure we've used for dealing with objects is not quite powerful enough for interaction. To see why this is so, take a look at the object tree on page 1575. You'll see that one of the fundamental design features of this tree is that each node branches in only two directions.

This feature is obviously very convenient, since each of the conditions we're testing at the choice nodes has only two possible values — true or false. Suppose, however, that the conditions were to have more than two possible values. If we wanted to check the value of the LCD flag (c$(n,9)), for example, which can have seven possible values (see page 1494) using a tree like the one on page 1596, we would need something like the structure shown in the first diagram. This is obviously rather cumbersome since it would be so much easier if we could use a structure like that shown in diagram 2.

In fact, we can do this quite easily, and still use the t array to hold data for our new tree. Our first tree used t(n,n,1) and t(n,n,2), which held the new



Multiple Choice

This tree shows how the program might process the Last Command Code flag (c$(c,9)) using a structure where each node can branch in more than two directions. This method enables more compact tree designs to be implemented in our program

node numbers that were to be branched to depending on whether the value held in the c array was one or two. We could, however, use t(n,n,1) to hold a *base* node number and t(n,n,2) to hold an *offset*. So, for example, node number one in the second diagram would have the value two read into the array element t(n,1,1), and the value of LCD minus one read into t(n,1,2). Using this method, we could traverse the tree using the formula:

New node = t(tree number, current node number, 2) + t(tree number, current node number, 1)

This is the method we'll use for our interaction tree. It has one or two limitations, but it lets us design a far more compact structure for the character handler. A provisional tree structure using this method is shown in the third diagram. First, it checks the strength of the character, since if a character is dead or unconscious, it obviously isn't going to do anything other than act as a focus for the attention of others. The program then decides whether or not to move a character by checking (and updating if necessary) the 'move' flag held in c$(n,11). Finally, one of three sub-trees is chosen — interaction with characters, object awareness, and activity reports. In the next instalment, we'll discuss these sub-trees in greater detail, together with the plot routines, and enter them into our program.

Basic Flavours

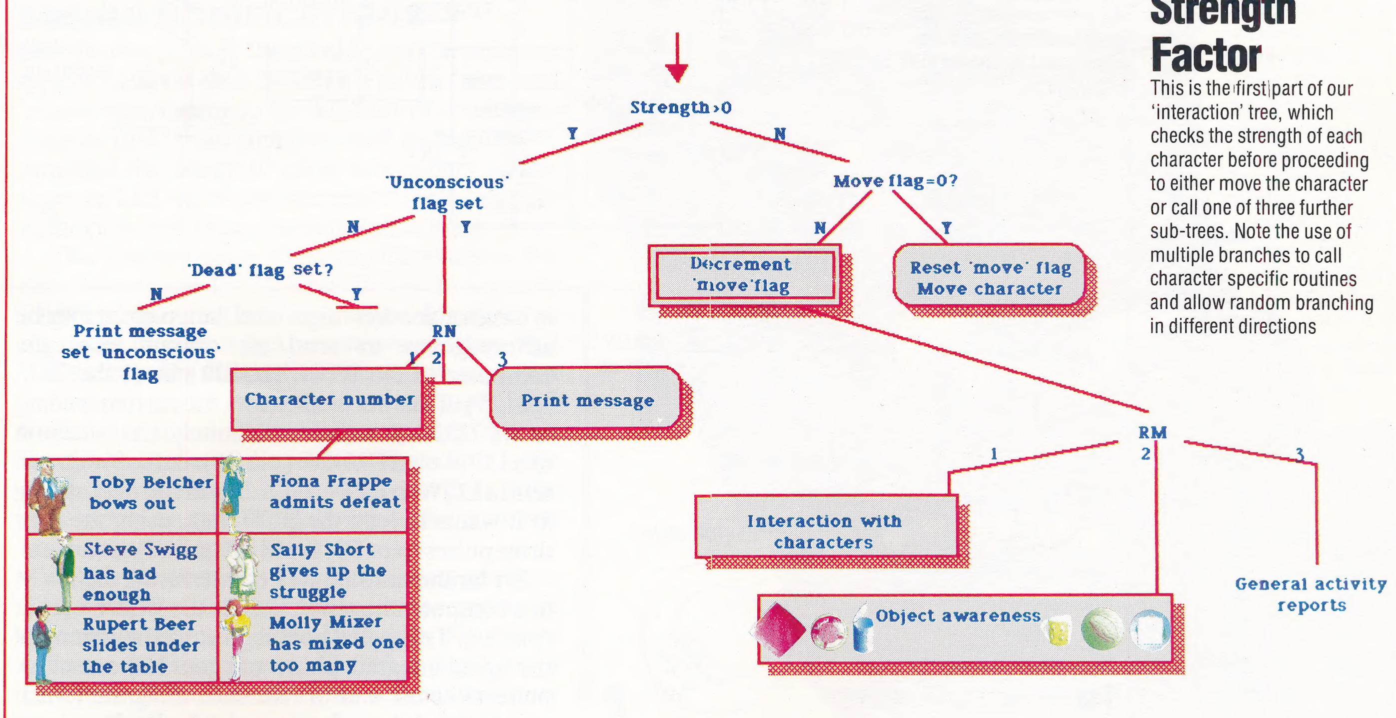
The following changes and additions should be made to the listing given in the previous instalment

Spectrum:

```
4180 q=INT(RND*2)+1: RETURN
5080 IF n=23 THEN GOSUB 2540: GOTO 5040
5085 GOSUB 2640: GOTO 5040
5090 RESTORE 9900: FOR e=1 TO (n-23): READ h:
NEXT e: GOTO h
9900 DATA 5100, 5130, 5160, 5180, 5210, 5240,
5260, 5270, 5280, 5300, 5310, 5330, 5340, 5360,
5370, 5430
```

BBC Micro:

```
4180 q=RND(2): RETURN
```



Strength Factor

This is the first part of our 'interaction' tree, which checks the strength of each character before proceeding to either move the character or call one of three further sub-trees. Note the use of multiple branches to call character specific routines and allow random branching in different directions

CAROLINE CLAYTON

METER MADE

We conclude our DVM project by looking at two optional extensions to the basic voltmeter design — a digital temperature probe and a digital computer interface. Both can easily be incorporated to produce a highly specified meter at a much lower cost than similar commercial models.

The easiest way of measuring temperature with a DVM is to buy a ready-made temperature probe that produces a voltage output proportional to the measured temperature. They usually cost between £20 and £30, however, and similar devices can be easily constructed for a fraction of the cost. Two possible circuits are given in diagram 1. The first, simpler circuit, produces a voltage output that increases by 10 mv per °C, but constant voltage will have to be subtracte from the output to obtain the necessary reading. If the DVM is connected to a microcomputer, this will be no problem as the subtraction can be done in software. The usual range is from −10° to +100°C.

The second circuit uses the temperature characteristics of a readily available BC109 PNP transistor buffered by an 'op amp'. This circuit will produce a linear output of from 0v to +5v for the temperature range 0° to 100°C.

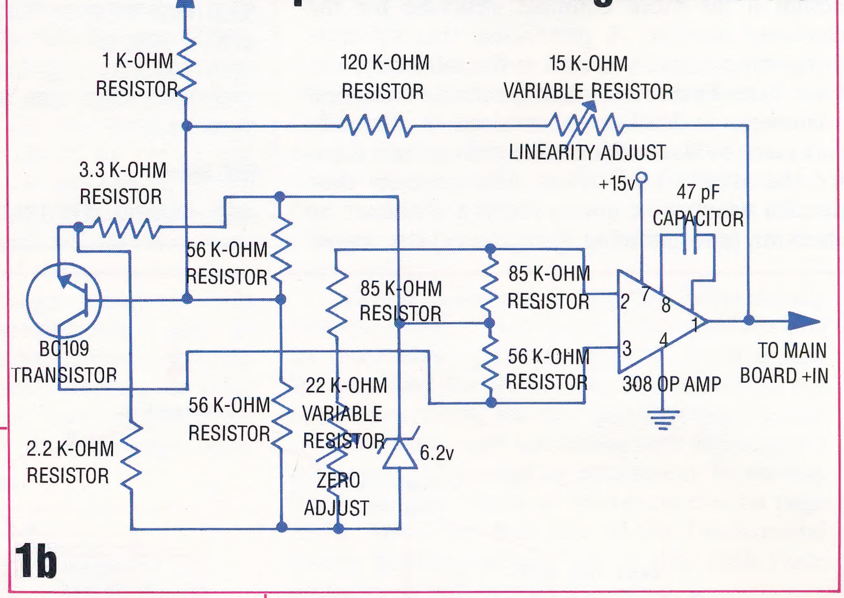
INTERFACING DETAILS

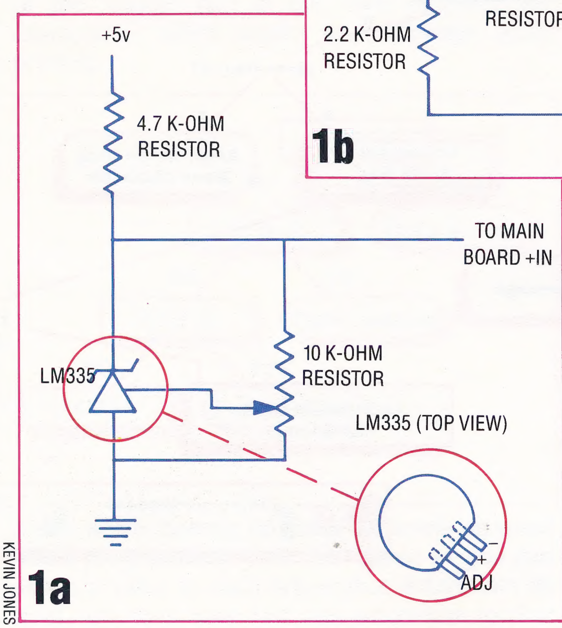
The 7135 A/D converter chip has been specially designed to make interfacing to a microprocessor as easy as possible, and there are a number of ways in which this can be done. First, the digital outputs from the chip are in BCD (binary coded decimal). Separate 'digit' outputs go true to indicate which digit the BCD data is valid for. There is a STROBE output (negative going) that can be used to latch the BCD data into a UART. The STROBE pulse goes negative momentarily once for each digit (five times during each measuring cycle). The pulse occurs at the centre of each digit drive TRUE pulse.

If the computer data-reading software is written in BASIC or another 'high-level' language, it may be advantageous to send an output from the computer to the RUN/HOLD pin of the 7135 (pin 25). If this pin is taken low, the current reading of the 7135 will be held indefinitely. On detection of a LOW on STROBE (pin 26), the software will send a LOW to RUN/HOLD and can take as long as it wants to read the BCD data, using the digit drive pulses (pins 20, 19, 18, 17 and 12) as strobes.

By far the simplest way of interfacing the DVM to a computer, however, utilises the BUSY output (pin 21). This signal goes high at the beginning of the signal integrate phase and goes low one clock pulse after the end of reference integrate (when the digital data outputs are latched). The signal

Temperature-To-Voltage Conversion





Temperature-To-Voltage Conversion
These diagrams show two alternative temperature-to-voltage conversion circuits. Circuit **b** is a more sophisticated circuit than circuit **a**, producing a change in voltage that is linearly proportional to the change in temperature. Two variable resistors allow the linearity and zero points to be accurately set

integrate phase takes 10,000 clock pulses (and BUSY goes low at the end of the first pulse after zero crossing). ANDing the CLOCK with BUSY and counting the number of pulses transmitted to the computer by the AND gate will therefore allow the measured voltage to be obtained by simply subtracting 10,001 in software.

The routine for counting the pulses will have to be written in machine code since BASIC would be too slow. At a clock frequency of 125 KHz the pulse will be positive for 4uS.

An even simpler alternative to counting the pulses would be a simple routine that waited for BUSY to go high and then loop until BUSY went low, adding one to COUNT each time round this loop. For this technique to work, the speed of the computer's CPU would have to be known accurately and the exact number of machine cycles required for each instruction in the machine code would need to be worked out. You would also need to have accurately measured the clock frequency of the DVM. This would allow you to measure the duration of BUSY quite accurately, subtracting 10,001 times the period of CLOCK to derive the duration of the reference integrate phase, and divide this result by the period of CLOCK to derive the number of CLOCK pulses in the reference integrate. Each pulse counted thus corresponds to 0.0001v.

The ANDing of the BUSY and CLOCK signals from the 7135 chip can be easily done using a 74LS00 chip, which has four NAND gates. Figure 2 shows the necessary pin connections to produce the BUSY.CLOCK signal. This chip can also be used to provide a buffered BUSY only signal if you wish to adopt the simple approach to measuring the duration of the reference integrate phase just outlined.

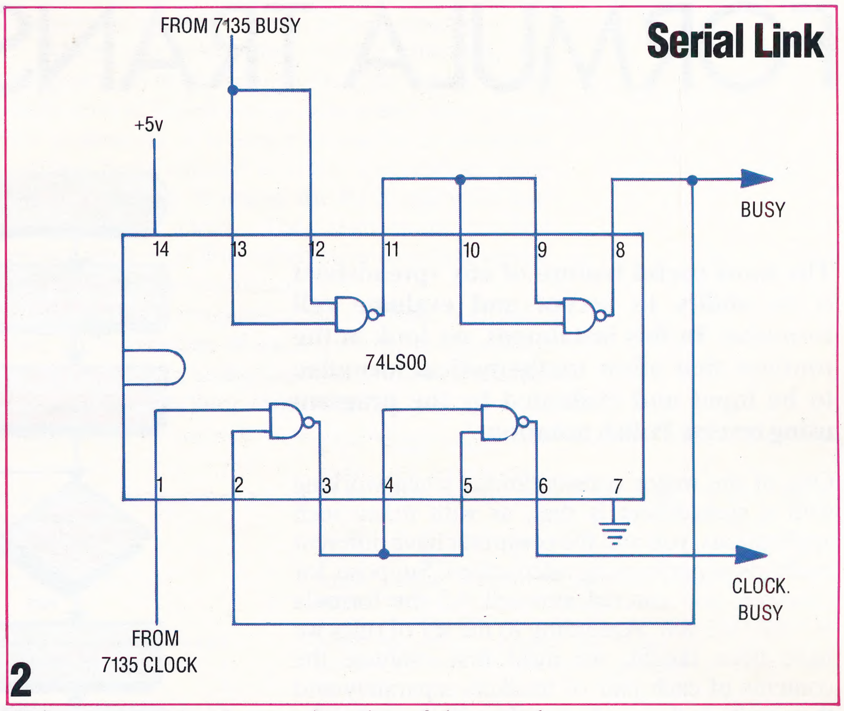
DESIGN MODIFICATIONS

Our prototype for the DVM used a different type of LED for digit 5, with a separate 'plus' and 'minus' sign. Owing to the difficulty of obtaining a suitable LED from suppliers such as Maplin, we modified the design to allow an ordinary seven-segment LED to be used for digit 5 (with segment g, the cross-bar, being used as a minus sign).

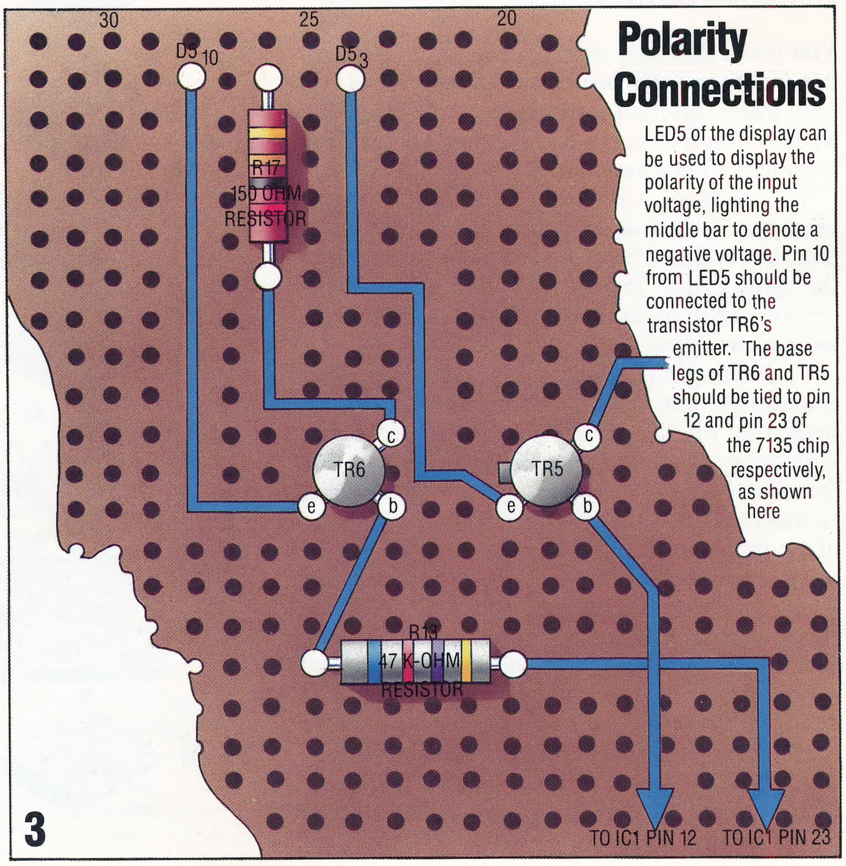
This involves some slight modifications to the circuit (see page 1553). Since the seven-segment LED has only one common anode for all seven segments, the minus sign can only be on when digit 5 is entirely lit (the previous arrangement had a separate anode for the sign). In order to use LED5 to show a minus sign, the following amendments should be made to the layout diagram on page 1553. TR5 is left as it was; the collector goes to the +5v supply and the emitter goes to the common anode of LED5 (pin 3).

TR6 is connected as follows: its base goes to pin 23 (polarity) of IC1, its emitter is connected to the cathode of segment g of LED5 (pin 10) and its collector is connected to digital ground via a 150ohm resistor. On the constructional diagram, the current limiting resistor from the collector of TR6 to ground is taken to the edge of the board (column 30). Don't forget to connect this point to the digital ground.

This then concludes our multimeter project. You have been given complete constructional details for building a basic, 2v sensitivity meter, together with moderately full details for a number of add-ons and a simple way of interfacing the meter to any computer. The style of housing for the unit, however, is left entirely up to you.



Serial Link
The signals from the 7135 A/D converter chip can be interfaced via a 74LS00 chip to produce a CLOCK.BUSY output. By connecting this output to a computer and counting the number of pulses received through software it is possible to calculate the voltage input to the voltmeter



Polarity Connections
LED5 of the display can be used to display the polarity of the input voltage, lighting the middle bar to denote a negative voltage. Pin 10 from LED5 should be connected to the transistor TR6's emitter. The base legs of TR6 and TR5 should be tied to pin 12 and pin 23 of the 7135 chip respectively, as shown here

FORMULA TRANSLATION

The most useful feature of any spreadsheet is its ability to accept and evaluate cell formulae. In this instalment, we look at the routines that allow mathematical formulae to be input and evaluated by the program using reverse Polish notation.

One of the major considerations when working with a spreadsheet is that, as with many such applications, you and the computer have different methods of performing calculations. Suppose, for example, you entered into cell A5 the formula (A1+A2)*(A3–A4). According to the set of rules we have been taught, we must first evaluate the contents of each pair of brackets separately and then multiply the results together. If we wanted the computer to perform this evaluation, we would have to instruct it to do the following:

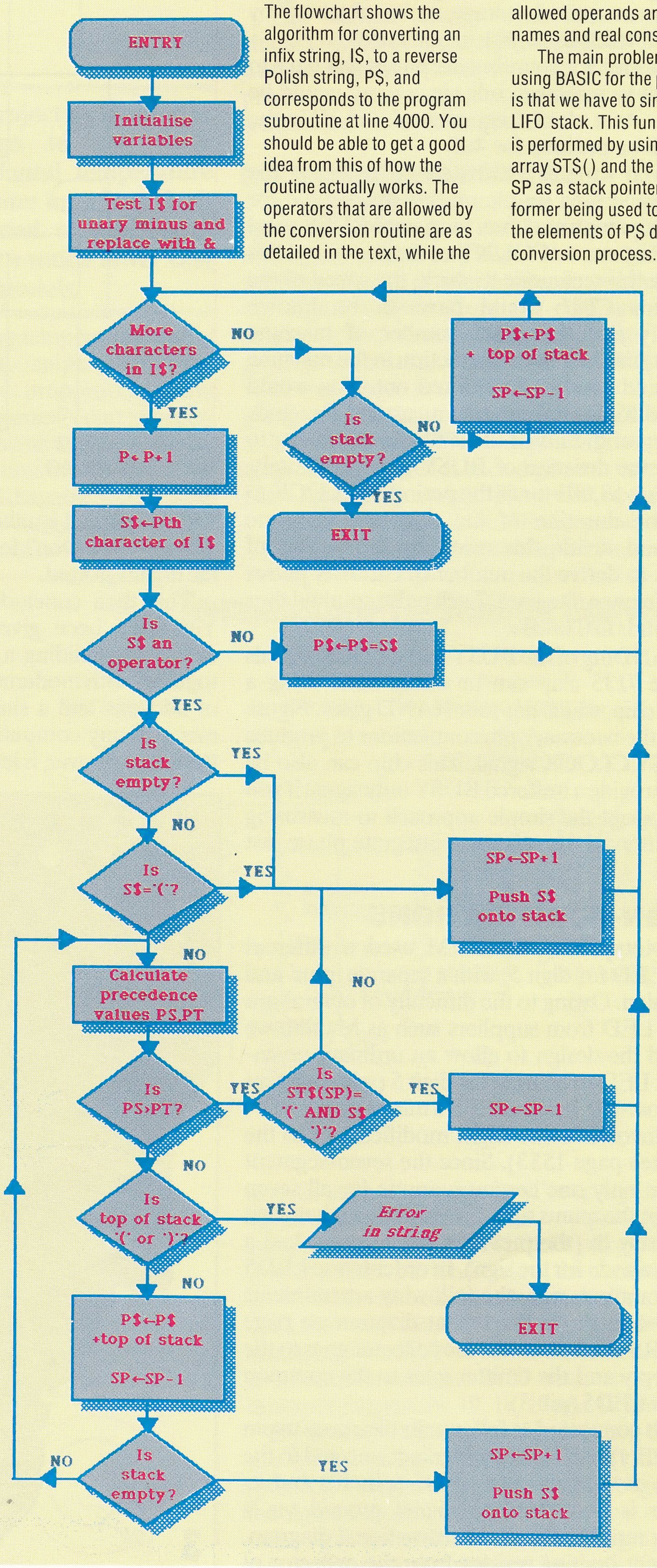
1. Get the values from cells A1 and A2.
2. Add them together and store the result termporarily.
3. Get the values from cells A3 and A4.
4. Subtract the value of A4 from A3.
5. Multiply the result by the result of A1 plus A2.
6. Store the result in cell A5.

You could probably write a program to perform this function, but it won't be of much use if you have many different formulae to evaluate. What we need, therefore, is a routine that will take any formula and evaluate it in the correct order.

The problem with writing expressions in the usual way (which is known as *infix* notation) is that they will not be evaluated by a computer in the same order. Instead, we use the system of reverse Polish notation (RPN; see page 1489) to write expressions.

Converting and evaluating the formulae in our spreadsheet can be divided into three distinct steps: step 1 is the actual process of converting the infix string to RPN; step 2 consists of checking that the RPN string produced is well formed (syntactically correct); and step 3 is the evaluation of the expression itself.

The operators allowed by this program are +, –, *, / and ↑. If you use the – sign in a formula to signify a negative value, the program requires the & character to differentiate this from the normal minus sign. To carry out the conversion from infix to RPN, we need to assign precedence values to each of the operators. This determines the order in which the operators are worked on. Division and multiplication, for example, are performed before addition and subtraction:



The flowchart shows the algorithm for converting an infix string, I$, to a reverse Polish string, P$, and corresponds to the program subroutine at line 4000. You should be able to get a good idea from this of how the routine actually works. The operators that are allowed by the conversion routine are as detailed in the text, while the allowed operands are the cell names and real constants.

The main problem with using BASIC for the program is that we have to simulate a LIFO stack. This function is performed by using the array ST$() and the variable SP as a stack pointer, with the former being used to stack the elements of P$ during the conversion process.

Operator	Precedence Value
=	0
(	1
+ −)	2
* / &	3
↑	4

In the final version of the spreadsheet program, the routine to convert infix formulae to reverse Polish will fit in as follows. After you input a formula that relates a number of cells mathematically, the resultant infix string is placed in an array, F$(), that stores the formulae for each cell on the sheet. The corresponding element in a second array, PS$(), which is used to hold the corresponding RPN string, is then cleared.

When you select the calculate function, the program will work through the PS$() array, taking each reverse Polish string in turn and setting it up for evaluation. If a new formula for a particular cell has been entered into the spreadsheet since the last calculate function, the RPN entry for that cell will be clear. The conversion routine presented here is thus called prior to setting up the cell formula for evaluation.

Using this arrangement, we can hold both infix and RPN versions of each formula in the sheet, using the former to display on screen and the latter for simple computer evaluation.

String Conversion

Basic Flavours

BBC Micro:

Make the following alterations to the Commodore 64 version:

Omit line 50

4240 IF ST$(SP)="(" OR ST$(SP)=")" THEN PRINT TAB(0,22);" ERROR IN FORMULA" : RETURN

Amstrad CPC 464/664:

Make the following alterations to the Commodore 64 version:

Omit line 50

4240IF ST$(SP)="(" OR ST$(SP)=")" THEN LOCATE 1,22:PRINT" ERROR IN FORMULA" : RETURN

Line 3140 should be renumbered as 3160

Commodore 64:

```
50 CD$=CHR$(17):CU$=CHR$(145):CR$=CHR$(2
9):CL$=CHR$(157):CO$=CHR$(19)
```

Addition To Arrays Subroutine

```
3140 DIM F$(255):DIM PS$(255)
```

RPN Conversion Routine

```
4000 REM ******************************
4001 REM * NORMAL TO REVERSE POLISH   *
4002 REM * STRING CONVERSION          *
4003 REM ******************************
4005 FOR K=1 TO 20:ST$(K)=" ":NEXT K
4006 LET P=0:LET SP=0:P$=""
4010 LET I$=C$
4015 FOR K=1 TO LEN(I$)-1
4017 J$=MID$(I$,K,1):K$=MID$(I$,K+1,1)
4020 IF J$="(" AND K$="-" THEN I$=MID$(I
$,1,K)+"&"+MID$(I$,K+2)
4025 NEXT K
4030 IF P<=LEN(I$) THEN 4100
4040 REM *** EMPTY STACK *****
4050 IF SP=0 THEN PS$((J-1)*15+I)=P$:GOS
UB 4700:RETURN
4060 IF ST$(SP)="(" OR ST$(SP)=")" THEN
4080
4070 LET P$=P$+ST$(SP)
4080 LET SP=SP-1
4090 GOTO 4050
4095 STOP
4100 LET P=P+1
4110 LET S$=MID$(I$,P,1)
4120 IF S$="+" OR S$="-" OR S$="*" OR S$
="/" THEN 4200
4130 IF S$="^" OR S$="(" OR S$=")" OR S$
="&" THEN 4200
4140 LET P$=P$+S$
4150 GOTO 4030: REM CONTINUE TO END OF S
TRING
4200 IF SP=0 THEN 4400
4210 IF S$="(" THEN 4400
4220 GOSUB 4500:REM ** PRECEDENCE **
4230 IF PS>PT THEN 4300: REM PRECEDENCE
4240 IFST$(SP)="("ORST$(SP)=")"THENGOSUB
1950:PRINT"   ERROR IN FORMULA ":RETURN
4250 LET P$=P$+ST$(SP):ST$(SP)="":LET SP
=SP-1
4260 IF SP>0 THEN 4220
4270 SP=SP+1
4280 LET ST$(SP)=S$
4290 GOTO 4030: REM CONTINUE TO END OF S
TRING
4300 IF ST$(SP)="(" AND S$=")" THEN LET
ST$(SP)="":SP=SP-1:GOTO 4030
4400 LET SP=SP+1
4410 LET ST$(SP)=S$
4420 GOTO 4030
4500 REM *** PRECEDENCE EVALUATION ***
4510 LET T$=S$:GOSUB 4600
4520 LET PS=T
4530 LET T$=ST$(SP):GOSUB 4600
4540 LET PT=T
4550 RETURN
4600 IF T$="=" THEN T=0:RETURN
4605 IF T$="(" THEN T=1:RETURN
4610 IF T$="+" OR T$="-" OR T$=")" THEN
T=2:RETURN
4620 IF T$="*" OR T$="/" OR T$="&" THEN
T=3:RETURN
4630 IF T$="^" THEN T=4:RETURN
4650 T=0:RETURN
```

Sinclair Spectrum:

Addition To Arrays Subroutine

```
3150 DIM F$(255,20):DIM H$(255,2
0):RETURN
```

RPN Conversion Routine

```
4000>REM ***********************
4001  REM * NORMAL TO REVERSE
*
4002 REM * POLISH STRING CONV. *
4003 REM ***********************
4005 DIM S$(20)
4006 LET P=0: LET SP=0: LET P$="
"
4010 LET I$=C$
4011 FOR Z=1 TO LEN (I$)
4012 IF I$(Z)=" " THEN  GO TO 40
14
4013 NEXT Z
4014 LET I$=I$( TO Z-1)
4015 FOR K=1 TO LEN (I$)-1
4017 LET J$=I$(K): LET K$=I$(K+1
)
4020 IF J$="(" AND K$="-" THEN
LET I$=I$(1 TO K)+"&"+I$(K+2 TO
)
4025 NEXT K
4030 IF P<LEN (I$) THEN  GO TO 4
100
4040 REM **** EMPTY STACK ****
4050 IF SP=0 THEN  LET H$((J-1)*
15+I,1 TO )=P$: GO SUB 4700: RET
URN
4060 IF S$(SP)="(" OR S$=")" THE
N  GO TO 4080
4070 LET P$=P$+S$(SP)
4080 LET SP=SP-1
4090 GO TO 4050
4095 STOP
4100 LET P=P+1
4110 LET O$=I$(P)
4120 IF O$="+" OR O$="-" OR O$="
*" OR O$="/" THEN  GO TO 4200
4125 IF O$="^" OR O$="(" OR O$="
)" OR O$="&" THEN  GO TO 4200
4140 LET P$=P$+O$
4150 GO TO 4030
4200 IF SP=0 THEN  GO TO 4400
4210 IF O$="(" THEN  GO TO 4400
4220 GO SUB 4500: REM PRECEDENCE
4230 IF PS>PT THEN  GO TO 4300
4240 IF S$(SP)="(" OR S$(SP)=")"
 THEN  PRINT AT 20,0;"ERROR IN F
ORMULA": RETURN
4250 LET P$=P$+S$(SP): LET SP=SP
-1
4260 IF SP>0 THEN  GO TO 4220
4270 LET SP=SP+1
4280 LET S$(SP)=O$
4290 GO TO 4030: REM CONTINUE TO
 END OF STRING
4300 IF S$(SP)="(" AND O$=")" TH
EN  LET S$(SP)="": LET SP=SP-1:
GO TO 4030
4400 LET SP=SP+1
4410 LET S$(SP)=O$
4420 GO TO 4030
4500 REM * PRECEDENCE EVALUATION
 *
4510 LET T$=O$: GO SUB 4600
4520 LET PS=T
4530 LET T$=S$(SP): GO SUB 4600
4540 LET PT=T
4550 RETURN
4600 IF T$="=" THEN  LET T=0: RE
TURN
4610 IF T$="(" THEN  LET T=1: RE
TURN
4620 IF T$="+" OR T$="-" OR T$="
)" THEN  LET T=2: RETURN
4630 IF T$="*" OR T$="/" OR T$="
&" THEN  LET T=3: RETURN
4640 IF T$="^" THEN  LET T=4: RE
TURN
4650 LET T=0: RETURN
```

THE JET SET

THE JET SET
THE JET SET

Entering The Jet Age
The Epson SQ-2000 is a high-quality ink jet printer that features a wide range of print styles and high-resolution graphics printing. The printer features Epson's new ink cartridge and automatic cleaning system

Room For Manoeuvre
The SQ-2000 doesn't have a computer interface included. Instead, one of three standard interfaces — Centronics, RS232C and IEEE — can be connected at the back of the machine. An optional 32K buffer and extra type founts in ROM can also be added on here

THE JET SET

CRISPIN THOMAS

Epson has established itself as one of the leading manufacturers of dot matrix printers for microcomputers. We take a look at the Epson SQ-2000 ink jet printer, the flagship of the company's new range of low cost, high reliability printers.

In the past few years, the average price of high-quality dot matrix printers has declined rapidly to fall within the range of many home computer owners. Epson's new range of printers seems set to do the same for ink jet technology. The SQ-2000, featured here, retails at around £1,800 but it is likely that Epson will introduce a uni-directional A4 size ink-jet printer — to be known as the HS-80 — for under £400. This will bring ink jet printers well within the reach of serious home computer users.

In common with its familiar relations — the Epson FX and MX range of dot matrix printers — the SQ-2000 is housed in a pleasing cream-coloured plastic casing. The control buttons, power switch and ink cartridge chamber are all easily accessible when the machine is resting on a desktop. As it is primarily designed for small businesses, the SQ-2000 has a wide carriage to accommodate special accounting and spreadsheet stationery. Epson has deliberately made its new printer as flexible as possible. Paper can be fed via the usual platen roller and sheet guide method, but optional cut sheet paper and continuous tractor feeders can be purchased as extras. The basic unit does not have a computer interface built in but there is a slot in the back of the machine for you to fit an interface cartridge. You can choose from three different standard interface types — Centronics parallel, RS232C serial or IEEE 488 — to suit the type of computer that the unit is to be used with. The interface cartridge is simply installed by pushing it home and securing it with a couple of screws. The standard interfaces have a two-Kbyte input buffer in which to store temporarily incoming data from the computer. An internal double-print buffer allows the simultaneous decoding and printing of data taken from the input buffer and a five-Kbyte download character buffer is available to store user-defined characters.

INK JET TECHNOLOGY

Ink jet printers have many advantages over dot matrix (see page 344) and daisy wheel (see page 364) printers. They combine the print quality associated with daisy wheels with the speed and flexibility of dot matrix printers, and without the noise level of either. In fact, the SQ-2000 is considerably quieter than any dot matrix or daisy wheel printer, which — as anyone who has to work in the same room as a printer will tell you — is a significant advantage. However, there have been problems with ink jet printers that have only recently been solved: the ink chambers were inclined to get blocked and the machines were

generally messy to operate. Epson claims that its new ink delivery system avoids these problems. In this system, the ink is delivered from a hermetically sealed ink cartridge that comprises three chambers, containing a specially formulated ink, cleaning fluid and storage space for waste products. The cartridge is designed to print over three million characters before needing replacement.

The reason underlying Epson's claim for improved reliability is that the ink delivery system is automatically cleaned (an operation taking only a few seconds) whenever the machine is switched on or off, and periodically while the machine is working. This cleaning operation can also be started manually by holding down a button on the front control panel. Other controls on this panel include the familiar on-line, line and form feed buttons, as well as a sheet feed button that allows you to load a single sheet from a bail. Three green indicator lights tell you that the power is on, whether the machine is on- or off-line and if it is ready to receive data from the host computer. Two red lights signal that you are low on ink or if the paper has run out.

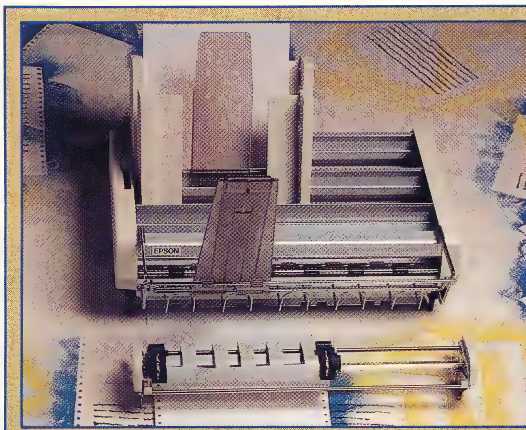
One of the most impressive features of the SQ-2000 is its range of printing styles. Two basic modes are available: draft and near letter quality (NLQ). Draft mode is faster than NLQ but has a commensurate loss of print quality. In each mode Pica, Elite and Roman founts can be selected in enlarged, condensed, italic, underlined, emphasised, proportional and super/subscript forms.

Like those produced by a dot matrix printer, the characters from an ink jet printer are formed from individual dots, allowing the flexibility of programmable character shapes. However, because the individual ink dots merge together, the resulting character has a better form than its dot matrix equivalent. The SQ-2000 produces its characters from a rectangular grid. In draft mode the characters are made up from a 15 by 24 dot grid; in NLQ mode the grid maintains its original size but has 29 by 24 dots packed into it to improve the quality of the print image.

The print head consists of 24 jets arranged in two rows of 12. The rows are offset slightly so that each jet overlaps sightly with its neighbour in the other row, and it is this overlap that gives the stronger print image. Like its dot matrix relations, the SQ-2000 can also be used to produce bit-mapped images, allowing high resolution screen displays to be dumped directly to it. The printing speed is a healthy 176 characters per second (cps) in draft mode and 105 cps in NLQ mode.

In addition to the cut sheet and tractor feeder add-ons, a number of other extras are available. These incude a 32-Kbyte input buffer that allows the host computer to rapidly spool out the data to be printed and then get on with other tasks while the data is emptied from the buffer and printed automatically. In addition to the founts available on the standard machine a number of other fount ROMs can be added.

This printer is an ideal purchase for the serious micro user as it is capable of performing all the major printing tasks demanded in a small business environment. Especially welcome is the reduction in noise level. Although the price tag of around £2,000 (including some of the optional extras) makes it a little out of the range of most home computer users' pockets, the proposed introduction of a slimmed-down version is extremely appealing.



Paper Chase
The SQ-2000 comes with a single sheet feeder but Epson has also made a tractor unit and bin feeder available as optional extras. The tractor unit has the familiar sprockets that pull continuous stationery past the print head. The bin feeder unit will be of great use to small businesses as it allows single sheets of paper to be fed continuously through the machine from two independently selectable bins — ideal for personalised letters or reports

EPSON SQ-2000

PRICE
Basic model: £1,825; 32K buffer: £99; tractor feed mechanism: £60

DIMENSIONS
620x297x188mm

INTERFACES
Centronics parallel, RS232C, IEEE 488

SPEED
176 cps in draft mode, 105 cps in NLQ mode

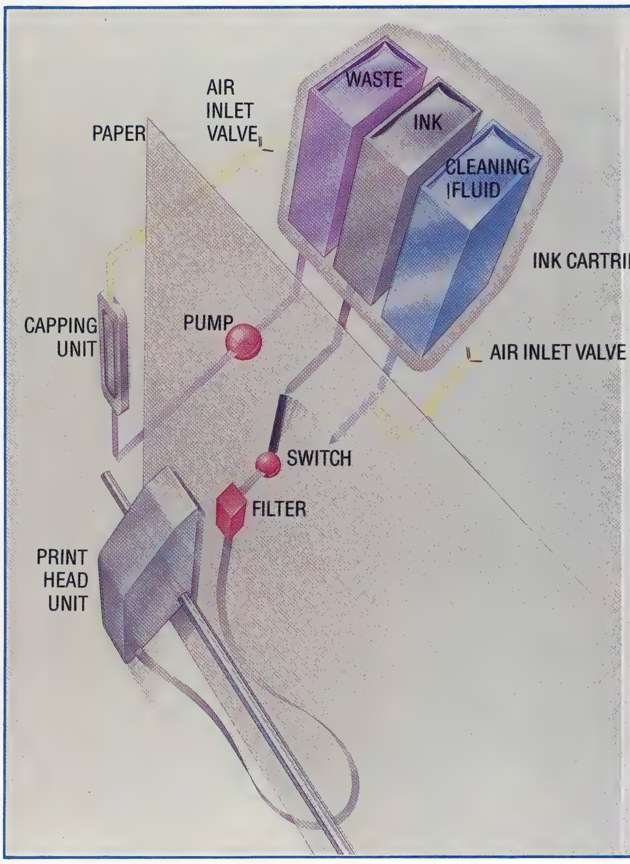
PAPER FEED
Friction or tractor

CARRIAGE WIDTH
Min 140mm, max 406mm

DOCUMENTATION
Well-written, concise manual with worked examples of user-defined character generation and bit-mapped graphics, full control code and character set appendices

STRENGTHS
Combines the flexibility and speed of dot matrix printers with the print quality of daisy-wheel types and significantly quieter than either

WEAKNESSES
Paper type must be carefully chosen to avoid smudging of the print



Keeping Clean

The new ink management system developed by Epson has three reservoirs to hold ink, cleaning fluid and waste products generated by the self-cleaning process. Pumping the cleaning fluid around the system keeps the pipes clear of possible blockages and a small rubber belt located on the left of the platen wipes the ink jet nozzles on the front of the print head

COURTESY OF EPSON LTD.

S

SHELL SORT

This is a method of data sorting devised in 1959 by the Dutch mathematician Donald Shell (hence the alternative name 'Shell's method'). The principle behind a *shell sort* is that it is faster to sort items by single long 'jumps' to their approximate position than by many short movements, which is typified by the 'bubble sort' method.

Assuming, for example, that we have 20 data items to be sorted, we first divide that number by two, giving 10. We can now count from the top, assigning a chain to each of the first 10 figures in the sort. The second group of 10 figures are then assigned to the chains so that each of the 10 chains now has two members. Next, each of the chains in turn is sorted — for instance, if chain 1 has a 9 as its first member and 4 as its second, their positions will be exchanged.

On moving to chain 2, we may find that it has a 3 as the first member and 7 as the second, in which case the figures will remain where they are. Once we have sorted the 10 chains, we can now halve the number of chains repeatedly until there is only one chain left, which will contain all the data items in order.

SHIFT REGISTER

This is a register within a computer that allows binary digits to be inserted from either end and will shift each of the other digits in the register by one position. *Shift registers* are often used in converting serial signals to parallel ones by accepting the incoming bits one at a time, placing them in the register which, when full, is sent along a parallel data bus or vice versa. However, all this is usually performed automatically from hardware and is of little interest to programmers. Where shift registers are of interest to this group is when they are used to perform bit manipulations on particular memory locations as part of a machine code program.

If we are dealing with a binary number in a register, shifting the number one place to the right performs a division by two. Similarly, shifting the number one place to the left multiplies the number by two. However, when dealing with negative numbers in two's complement (see page 328) format, arithmetic meaning is only preserved during a right shift if a one is inserted at the left-hand end. Many instruction sets therefore differentiate between arithmetic and logical shifts. While the former preserves arithmetic meaning (a register containing -16 will contain -8 after a right shift), the latter only preserves the bit pattern, by inserting a zero in the vacant bit position.

SIMULATION

This is a computer program that aims to emulate a series of events that might occur in the real world. The intention of such a program is to allow the user to investigate the effects of particular courses of action, the purpose of which is usually twofold. It can be used on the one hand to design an object without the expense of having to build and test it under the conditions in which it will be used. A programmer could, for example, provide a *simulation* that tested the aerodynamic effects on a particular design of car without actually having to build it and place it in a wind tunnel.

The other purpose of simulations is to provide training without the risk of inexperienced users potentially damaging highly expensive pieces of equipment or causing injury to themselves or others. One of the best known examples of this is an aircraft simulator. Using a real aircraft could not only cost millions of pounds if the pupil made a mistake, but would prevent that aircraft from being used for more profitable uses, to say nothing of the price of fuel and other costs.

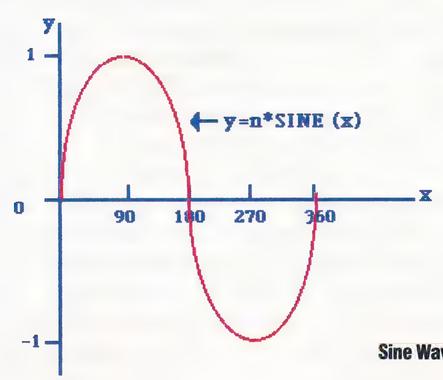
Cockpit To Computer
Flight simulators are available on a range of home micros and respond to keypresses or joystick movements from the user. The more sophisticated programs can not only simulate the flying of an aircraft but also weather conditions and the layout of airport runways

This is the essential use of simulations. A computer simulation costs far less than it would to use the real thing. However, in order for a simulation to be meaningful, it must replicate the conditions that would prevail in the real world as far as possible. Thus every condition that might have an effect in reality must be taken into account when designing a simulation.

SINE WAVE

A *sine wave* is a pure wave pattern that is described by the formula Y=N*SINE(X). Thus incrementing X will produce values of Y which, when plotted on a graph, will produce the characteristic sine curve.

Sine waves are often used for calibrating electronic devices. Because it is a pure waveform, the sine wave can be used as a reference against which other signals can be compared.



Sine Wave

CAROLINE CLAYTON

NUMBER CRUNCHER

Although FORTRAN first appeared in the mid 1950s, it still retains much of its initial usefulness and popularity, especially in numerical applications. In this first of three instalments on FORTRAN, we look at its general program layout as well as its I/O and arithmetic implementations.

FORTRAN is generally regarded as the first of the high-level languages of the type we are familiar with today. It was designed at a time when nearly all computers were being used for purely numerical work so that its facilities for handling character strings, for example, are very limited — but it still remains one of the few languages that can directly handle complex numbers. Its mathematical nature means that fast, efficient numerical routines can be written, making it a good language for graphics; there are also a number of machines whose operating systems have been written at least partly in FORTRAN.

Over the years, many subroutine libraries have been built up for a variety of numerical applications and this is one reason for the continuing popularity of what is, after all, a very old-fashioned language, since the easiest way to utilise this large body of software is to continue writing in FORTRAN. It would appear, then, that FORTRAN will continue to be the major programming tool in scientific laboratories and engineering for some time.

There are a number of FORTRAN versions available on microcomputers, though few of them are full implementations of the current standard, FORTRAN 77. Most are versions of the previous standard, FORTRAN IV, with reasonably consistent extensions. We'll concentrate on FORTRAN IV to start with, since this gives the best flavour of the language, then try to indicate the common extensions and relaxations from the standard, which most micro implementations allow.

The layout of a FORTRAN program is determined by the fact that most input/output used to be on 80-column punch cards. The last eight columns on such a card were normally reserved for a card identification number, which left 72 columns available for each program statement. Consequently, FORTRAN normally accepts only one statement per line and one line per statement with a maximum length of 72 characters. If it is necessary to spread a statement over more than one line, then the following lines have to be specially marked as continuation lines. The remaining 72 columns are further restricted as the first five columns are reserved for statement numbers, column six is used to indicate whether the line is a continuation or not and the actual statement begins in column seven.

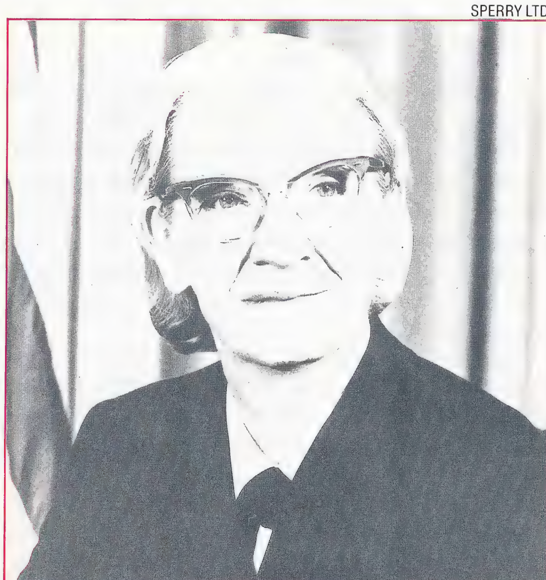
Lines don't have to be numbered as in a BASIC program, but any line that is referred to in another statement must be given a number as a label. These numbers don't have to be in any particular sequence, but must be unique. A line can be used as a comment by putting a C in column one. FORTRAN programs, like most BASIC programs, are difficult to structure properly and lacking in data types and control structures, making them difficult to read and therefore requiring comments as an important feature.

Changing Times

Chronology of FORTRAN versions:

1954	First specifications for FORTRAN issued
1957	First general release of compilers for FORTRAN (I)
	Specifications issued for FORTRAN II
1958	First FORTRAN II compilers available
1962	Specifications issued for FORTRAN IV
1963	Compilers available for FORTRAN IV
1977	Specifications for new FORTRAN 77
1978	First compilers available for FORTRAN 77

SPERRY LTD.



Grace Hopper
Grace Hopper was one of the driving forces behind the development of early programming languages. Working at Remington Rand in the mid-50s, she helped produce one of the first compilers – A-2 – and the subsequent languages ARITH-MATIC and FLOW-MATIC. The concept of compiled languages was then refined by IBM in their first version of FORTRAN, while FLOW-MATIC later developed into COBOL

As in BASIC, but not PASCAL, variables don't have to be declared; they may be introduced at any point in a program merely by mentioning a new name. If you do this, however, then you accept the default data typing, which is that any variable name beginning with I, J, K, L, M or N is integer, with anything else being real.

The complexities of handling input/output are normally a major difficulty for novice FORTRAN programmers. The problem arises from the basic reliance on punch cards with their strict format. Each I/O operation requires two statements — a READ or WRITE, which specifies both the I/O device and the variables being used; and a FORMAT statement, which specifies the exact types and layout of the data on the card, record or line of terminal input. For example:

```
    READ(1,100) X, I, ICHAR
100 FORMAT (F7.2, I4, A1)
```

The READ is the executable statement that will cause an input operation to be carried out. The first number inside the bracket specifies the input device being used. The designation of numbers to devices depends on the exact implementation and the hardware, but one number should refer to the terminal, others to the printer (for output only), card reader or card punch, or may be assigned to files on disk or tape.

The second number inside the bracket is a statement number identifying the associated FORMAT statement. (Notice that the FORMAT statement in the example has been given that same number.) The FORMAT statement is not executable; it simply gives information so it may be positioned anywhere in the program. The number is the only way of showing the association, and it is quite permissible for more than one READ or WRITE to refer to the same FORMAT. Following the bracket after the READ is the list of variables to which the input values are to be assigned.

The FORMAT includes a specification for each variable in the list. The main specifications are:

- F: F7.2, for example, indicates that the value to be input to X is real, occupying the first seven columns of the line, record or card, with two digits after the point.
- I: I4 indicates that the value to be input to I is an integer, occupying the next four columns on the line.
- A: A1 indicates that the value in the next column is a character.

The only way that FORTRAN can handle characters is to store them in integer variables. The maximum number per variable depends on the size allocated to the system for an integer variable, so using 16-bit integers means that only A1 and A2 are allowable. However, some versions allow characters to be stored in real variables as well. Since the variables can be treated as both numeric and alphabetic, and the way the characters are stored is not always in accordance with strict ASCII code, this can lead to a lot of confusion. Longer character strings, for instance, have to be handled using integer arrays. FORTRAN is therefore not a language to be recommended for applications involving the processing of text!

The actual values that are input must fit this specification, otherwise a run-time error will be generated. Most modern systems, however, will allow a so-called 'free-form' input, such as READ(1,*)A,B,C or just READ A,B,C, which is meant specifically for keyboard input where it is difficult to achieve the exact spacings between values.

Comparative Logic

Logical operators and connectives:

FORTRAN	BASIC
.EQ.	=
.NE.	<>
.GT.	>
.GE.	>=
.LT.	<
.LE.	<=
.NOT.	NOT
.AND.	AND
.OR.	OR

Good With Numbers

FORTRAN IV allows a number of numeric data types:

- INTEGER and REAL which have their usual meanings
- DOUBLE PRECISION which are floating point numbers using double the number of words of memory then ordinary reals
- COMPLEX for complex numbers with real and imaginary parts
- LOGICAL equivalent to a PASCAL boolean, which can take the values .TRUE. or .FALSE. (the surrounding dots are necessary)
- Variable names may be up to 6 characters long using only alphabetic or numeric characters, the first must be alphabetic
- Variables may be declared only at the start of the program by means of a declaration statement, such as:

```
INTEGER NUM1,NUM2
LOGICAL FNSH
```

These free-form inputs, then, function in a similar way to the BASIC INPUT statement. Output is handled in a similar way using a WRITE statement:

```
    WRITE(1,200) A,B,C
200 FORMAT(3F7.2)
```

Note how repeated specifications can be handled without having to write one for each variable.

Text strings can be output using a special H specification, but this is awkward to use as the exact number of characters must be stated:

```
      WRITE(1,300)ANS
300   FORMAT(13HTHE ANSWER IS,F7.2)
```

Again, many modern systems will allow quoted strings inside the FORMAT:

```
      WRITE(1,300)ANS
300   FORMAT('THE ANSWER IS',F7.2)
```

but this is not standard FORTRAN.

When the output is directed to a line printer, FORTRAN uses the first printed character for paper control, though the exact details may vary between installations. A typical system, for example, may use a space character for normal single spacing and the characters 1, 2, 3 and so on to leave the appropriate number of blank lines. A common error is to forget this and to lose the first character of your output while the printer does something unpredictable with the paper. A typical output to the printer might be:

```
      WRITE(2,400)I,J,K,L,M,N
400   FORMAT(1H,2(I6,5X))
```

Notice the extra 1H, which controls the paper. The X specification simply leaves the specified number of blank spaces. A sequence of specifications can be bracketed with a multiplier, as in the 2(I6,5X).

In the examples, we'll use a device number of 1 throughout to indicate input/output via a terminal, and the actual value to be used here will depend on the implementation being used.

FORTRAN is very similar to BASIC in the way that arithmetic and assignment are set out, and the only major difference is that FORTRAN uses a double asterisk (**) for exponentiation instead of the up-arrow (↑). Real, integer and even double precision values and variables can be mixed within an expression and the result assigned to a variable of either type. Care must be taken, however, otherwise you may find all or part of a calculation being performed in integer arithmetic when a real result is required:

```
I=2
J=3
A=I/J
```

A will have the value 0 as the division has been done in integer arithmetic. A FLOAT function is provided to convert an integer value to a real so that:

```
A=FLOAT(I)/FLOAT(J)
```

would work correctly.

Having been designed as a numeric language, you would expect to find in FORTRAN a large variety of standard functions. There are too many to list here, however, including all the usual trigonometric and logarithmic functions as well as many more that would only have meaning to mathematicans or engineers. But it's a fairly safe bet that any standard numerical operation is available as a FORTRAN function. Finally, most functions are provided in both real and double precision versions and, where appropriate, in an integer version as well.

One of the major criticisms of FORTRAN has been its lack of control structures, which has been remedied to a certain extent in FORTRAN 77. The familiar GOTO statement is used extensively and its destination can be any numbered executable statement (not a FORMAT statement).

There are two varieties of IF statement: The logical IF is similiar to the BASIC one, taking the form:

```
IF (logical expression) executable statement
```

(The logical operators and connectives are a different matter.) The other form of IF is the arithmetic type, which is often quoted as the worst example of a control structure. This takes the form:

```
IF (arithmetic expression) S1, S2, S3
```

where S1, S2 and S3 are statement numbers. Control is transferred to statement number S1 if the value of the arithmetic expression is less than zero, to S2 if it equals zero and to S3 if it is greater than zero. This can be quite convenient in the way that it allows a three-way branch, but it leads to even worse 'spaghetti code' than a plain GOTO.

Average Student

```
C     PROGRAM TO READ IN A SET OF NUMBERS
C     AND PRINT THEIR AVERAGE.
C     INPUT IS TERMINATED BY
C     ENTERING A NEGATIVE NUMBER.
C
C     INITIALISE COUNT AND SUM
C
      SUM=0
      ICOUNT=0
C
C     READ NEXT NUMBER
C
100   READ(1,10)X
C
C     CHECK IF NEGATIVE
C
      IF (X) 300, 200, 200
C
C     POSITIVE NUMBER
C
200   ICOUNT=ICOUNT+1
      SUM=SUM+X
      GOTO 100
C
C     END OF INPUT
C
300   AVGE=SUM/FLOAT(ICOUNT)
      WRITE(1,20)AVGE
      STOP
C
C     FORMATS
C
10    FORMAT(F9.2)
20    FORMAT (10HAVERAGE IS, F9.2)
      END
```

ON DISPLAY

The Amstrad operating system supports a character-based text display that is entirely separate from the graphics display. We look at aspects of screen control that are useful to the machine code programmer, and provide three listings that can be incorporated into your own programs.

Useful Addresses Table

Routine	Address	Notes
TXT_WIN_ENABLE	BB66	Enter with H-left; D-right; L-top; E-bottom
TXT_GET_WINDOW	BB69	Exit with registers as above
TXT_CLEAR_WINDOW	BB6C	Clears text window of current stream
TXT_SWAP_STREAMS	BBB7	B and C hold the two stream numbers
TXT_STR_SELECT	BBB4	Stream to be selected in A
TXT_OUTPUT	BB5A	A holds character
TXT_WR_CHAR	BB5D	A holds character
TXT_RD_CHAR	BB60	Reads character at cursor and returns it in A
TXT_GET_CURSOR	BB78	H gives column, L gives row
GRA_SET_ORIGIN	BBC9	Enter with DE-X, HL-Y co-ordinates
GRA_SET_WIDTH	BBCF	Enter with DE-left, HL-right co-ordinates
GRA_SET_HEIGHT	BBD2	Enter with DE-top, HL-bottom co-ordinates
SCR_SET_BASE	BC08	A holds hi-byte of base address
SCR_SET_MODE	BC0E	A holds mode number (0, 1 or 2)

Note: To make proper use of these routines, you will need to refer to the Amstrad Firmware Specification, published by Amsoft (SOFT 158), which contains full exit and entry conditions of these and all other firmware routines

The text VDU, among other things, manages the reading and writing of characters on the screen, the cursor, windows and the user-definable symbols. The Amstrad firmware allows up to eight streams and their associated windows to be in use anywhere on the screen, each with its own pen and paper inks, cursor, opaque or transparent background mode, and graphics character write option. The size of the stream windows can be set with TXT_WIN_ENABLE, interrogated with TXT_GET_WINDOW and cleared using TXT_CLEAR_WINDOW.

Additionally, any two streams can have all their parameters swapped with another by TXT_SWAP_STREAMS. Most of the firmware routines act upon the current stream; the entry TXT_STR_SELECT is provided to select the current stream, which remains in use until another stream is selected.

Each character on the screen is displayed within an eight-by-eight pixel square. There are several firmware entries that are used to print a character, the most significant of which are shown in the Useful Addresses table.

Normally, characters occupy adjacent squares and are printed at the current text cursor location. There is, however, a special mode — selected with TXT_SET_GRAPHIC — that permits characters to be plotted at the graphics cursor location, and this allows text to overlap and to appear at any position on the screen.

The main routine, TXT_OUTPUT, simply saves all the registers before calling the 'indirection', TXT_OUT_ACTION. The routine pointed to by the indirection deals with any control codes before passing displayable characters to TXT_WR_CHAR, or GRA_WR_CHAR if the graphics character write option is enabled. The indirection may be patched to provide an alternative means of interpreting control codes or writing characters to the screen.

The text VDU has an associated cursor that points to the position at which the next character is to be displayed. Routines are provided to enable and disable the text cursor and to place or remove another cursor at any character position. The most useful routines associated with cursor positioning are detailed in the table.

The Text Tab program allows tab stops to be set at equal distances according to the variable TAB. The current window size is first obtained with a CALL to TXT_GET_WINDOW; this returns the right-most column in register D, which is converted to point to one position after the right-hand column margin. Next, the actual position of the cursor is obtained with TXT_GET_CURSOR; register H returns the cursor column, which is successively

compared to every possible tab stop until the next tab stop is found. Once the next possible tab stop column is known, a test is done to ensure that there is enough space for the field of characters to follow after it. If there is room then the cursor column is set to correspond to the next tab stop, otherwise the cursor is forced to move to the next line by specifying a column outside of the current window. If necessary, the routine can be patched into TXT_OUTPUT to allow different tab stops, other than the standard eight columns, to be supported.

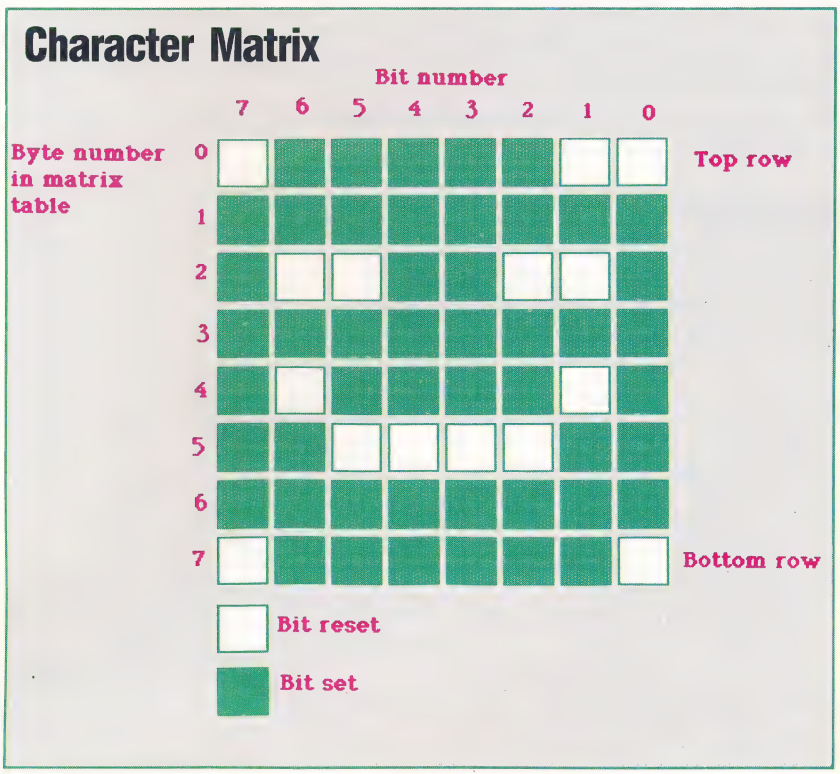
Pixel Assignment

Bit number	7	6	5	4	3	2	1	0
Mode 2	1	2	3	4	5	6	7	8
Mode 1	1	2	3	4	1	2	3	4
Mode 0	1	2	1	2	1	2	1	2

Mode Differences
Each byte in the screen memory holds data for between two and eight pixels, depending on the display mode. The diagram shows which bits in each byte refer to which pixel in different modes

USER-DEFINABLE CHARACTERS

Each of the ASCII codes from 0 to 255 has an associated matrix that defines which bits make up the corresponding displayed character. The matrix consists of eight bytes, where each byte makes up one line of the character. Our diagram shows how the character matrix for CHR$224 is stored in memory.



Quite A Character
Characters are stored in eight-byte matrices, with each byte representing one pixel row of the character. Set bits indicate a pixel to be printed in the current pen colour, reset bits indicate that the paper (background) colour should be used. Our diagram shows the matrix for ASCII character 224

There are four firmware routines that allow matrices to be determined and redefined. By default, all the matrices for ASCII characters 0 to 247 are held in the lower ROM and are therefore not initially redefinable; characters 248 to 255 are copied into RAM where they may be redefined. The routine TXT_SET_M_TABLE may be used to redefine any number of characters between the one specified and 255, thus allowing an entirely new character set to be used if desired.

Characters that have been allocated as redefinable can have their matrices manipulated directly by a user program, or may be set with the routine TXT_SET_MATRIX which takes eight bytes pointed to by register HL and copies them into the matrix corresponding to a given character.

A routine TXT_GET_MATRIX is also provided to obtain the address of a character matrix, and the routine TXT_GET_M_TABLE may be used to return both the first character from, and the address of, any user-definable table.

Our Spin Print program makes use of TXT_GET_MATRIX and TXT_SET_MATRIX to rotate characters in one of four directions. This is achieved by redefining character 248 to have the same matrix as the required character and then manipulating the matrix in character 248 for the desired rotation, before printing it with TXT_OUTPUT. Useful textual effects can be obtained by linking the program to BASIC.

THE GRAPHICS VDU

The graphics VDU is provided for plotting on the screen at pixel resolution. A single graphics window is used, in which the foreground and background inks may be set independently to those used for the text VDU.

There are three co-ordinate systems that can be used to describe a pixel position on the screen. The actual screen resolution is 640 by 200 dots in mode 2, 320 by 200 dots in mode 1 and 160 by 200 dots in mode 0; these are termed the *base co-ordinates*. Each point on the screen has, however, a unique address which is used to reference it regardless of the current mode. The different modes simply determine how much area around a particular pixel is also affected by setting that point. Effectively, one pixel is set in mode 2, two in mode 1 and four in mode 0.

Although the physical screen has only a resolution of 200 points vertically, this is treated as 400 points by the firmware so that an approximate aspect ratio of 2:1 can be maintained. Each vertical point, therefore, refers to half a pixel, and two halves of a pixel are always set at the same time. Pixel rows 10 and 11, for example, both reference the same line on the screen.

The screen is therefore always treated as having a resolution of 640 by 400 in any mode.

The co-ordinate system described above applies to the absolute positions on the whole screen. It is possible, however, to define a graphics window that uses a third set of co-ordinates to reference a point.

The window size is set using GRA_SET_ORIGIN, GRA_SET_WIDTH and GRA_SET_HEIGHT; and may be determined using the corresponding GRA_GET_ORIGIN, GRA_GET_W_WIDTH and GRA_GET_W_HEIGHT. Having set a window to

use, all co-ordinate references are made relative to the new origin. These co-ordinates are termed the *user co-ordinates*. The graphics cursor is positioned using one of two methods: *absolute* or *relative* positioning. If the absolute method is used, then the cursor is positioned at the user co-ordinate specified. If the relative method is used then the cursor is moved the number of points specified, relative to its current position. The entries used for cursor positioning are detailed in our table.

The graphics VDU allows pixels to be plotted in three ways — individually, as characters or as lines. All the plotting routines have the option of either plotting absolute points or points relative to the current graphics cursor position. The graphics cursor is always left at the last point plotted, except when plotting characters where the vertical position remains unchanged and the horizontal position is moved to the start of the next character. The relevant entries are detailed in the table.

THE SCREEN PACK

The *screen pack* is a section of the firmware that provides the lowest level access to the screen. It handles the screen addressing, palette setting, ink encoding and decoding, mode setting and screen scrolling, among other things.

The screen is a 16 Kbyte block of RAM that may be set to lie at any one of the four 16-Kbyte boundaries in memory. It is only useful to use the blocks starting at $4000 and $C000, as the other two blocks contain areas of the firmware that would become overwritten. The screen can be switched between the two useful blocks by using the SCR_SET_BASE entry; this technique can be used to prepare one screen while displaying another and then making the screen update appear instantaneous.

Each byte of memory used for the screen contains the information for eight physical locations on the screen. These may correspond to two, four or eight pixels depending on the mode selected using SCR_SET_MODE.

The Pixel Assignments diagram shows how each pixel in a byte is determined from the bits in the different modes. Although the system at first appears fairly difficult, it is in fact designed so that rotating a byte gives the bits required for the next pixel to the left or right. So, to set a pixel within a byte, a mask can be generated to set the left-most pixel and then rotated until it is in the appropriate position.

The fastest way to write to the screen is to access the screen memory rather than using the text or graphics VDUs. The screen pack provides several routines to make this a simpler task.

Finally, our Fast Point Plot program is an example of a listing that uses these entries to plot a physical point on the screen. Note that this cannot therefore be used in conjunction with the graphics screen if a window has been set up.

The routine first of all asks for the address of the byte containing the information for the specified screen position, and then for the mask to set all the pixels in that byte to the specified ink. The mask is then manipulated to reset the required position to ink zero and then set it to the new ink.

128K Plot

The Amstrad 6128 offers a slight refinement to the PLOT command as implemented on the CPC 464. The firmware on both models provides a routine to select one of four different plot modes. These are:

Mode	Action
0	Normal — set pixel to new ink
1	XOR — set pixel to (new ink XOR existing ink)
2	AND — set pixel to (new ink AND existing ink)
3	OR — set pixel to (new ink OR existing ink)

The XOR mode is particularly useful, since it enables you to plot lines and then erase them without disturbing the background colour if you so desire. For example, if your background colour is red (paper 3), then plotting a line in white (pen 4) will produce a magenta line (pen 7). This can be erased by plotting a second time in pen 4, simultaneously restoring the original background colour.

This mode of plotting is particularly useful in sprite management routines, which need to move over a backdrop without disturbing it. On the 6128, plotting modes are specified by adding an extra parameter to the PLOT command, but this facility is not provided in BASIC 1.0 on the CPC 464. All that is required, however, is a call to the firmware routine at &BC59, with the required mode number in A

Tab Setter

This program moves the text cursor for the currently active window to the next tab column, as set by the variable TAB. There are no entry conditions and the routine preserves all registers

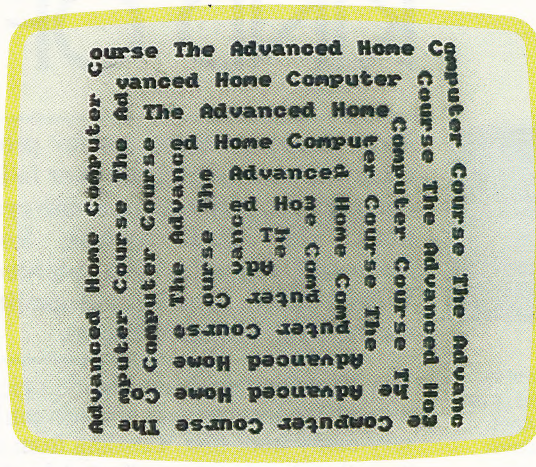
```
tab:            equ     15          ; tab stop column
get.window:     equ     #bb69       ; TXT_GET_WINDOW
get.cursor:     equ     #bb78       ; TXT_GET_CUSOR
set.column:     equ     #bb6f       ; TXT_SET_COLUMN
;
        push    af
        push    hl
        push    de
        call    get.window          ; get window size
        inc     d                   ; logical rt col
        inc     d                   ; right col limit
        call    get.cursor          ; get cursor posn
        ld      a, 1                ; start of line
;
loop:   add     tab                 ; get next tab col
        cp      h
        jp      p, loop
;
        ld      h, a                ; save it
        add     tab                 ; room for new col?
        cp      d                   ; test for limit
        jr      nc, setcol          ; no,force newline
        ld      a, h                ; get back new col
setcol:
        call    set.column          ; set new column
        pop     de
        pop     hl
        pop     af
        ret
```

Spin Print

This program prints a character after rotating the character matrix to one of four angles. The address of the character matrix is found and is then used to redefine character 248, which is normally the first entry of the user-defined character table. If this character is not redefinable then no action is taken

```
;
; Entry:        IX+0 = Character to display
;               IX+2 = Desired angle 1..4
;                  1 = upwards
;                  2 = backwards
;                  3 = downwards
;                  4 = normal
; Exit:
;               Carry True  - Character printed
;               Carry False - Failed to print char
;               AF, BC, DE, HL corrupt
;
txt.get.matrix: equ      #bba5
txt.set.matrix: equ      #bba8
txt.output:     equ      #bb5a
;
        org     8000h
;
start:  ld      a, (ix+0)        ; read character
        call    txt.get.matrix   ; get matrix addr
;
; Copy matrix into user definable character 128
; then find its address
;
        ld      a, 248           ; char to redefine
        call    txt.set.matrix   ; redefine matrix
        ret     nc               ; abort if unable
;
        ld      a, 248           ; char to rotate
        call    txt.get.matrix   ; find matrix
;
; Rotate the character matrix anti-clockwise
; in 90 degrees steps
;
        ld      (toprow), hl     ; save address
        ld      b, (ix+2)        ; read angle
;
lop3:   ld      d, 8             ; loop for 8 rows
;
lop2:   ld      e, 8             ; loop for 8 bits
;
lop1:   rlc     (hl)             ; next bit in c
        rla                      ; save in new row
        inc     hl               ; go to next row
        dec     e
        jr      nz, lop1         ; go for next row
;
        push    af               ; stack new row
        dec     d
        ld      hl, (toprow)     ; go for next bit
        jr      nz, lop2
;
; loop to unstack the new matrix
;
        ld      e, 8
lop4:   pop     af               ; get next row
        ld      (hl), a          ; move to matrix
        inc     h                ; go to next row
        dec     e
        jr      nz, lop4
;
        ld      hl, (toprow)
        djnz    lop3             ; again if needed
;
; Finally display the character on screen
;
        ld      a, 248           ; char to print
        call    txt.output
        ret                      ; finished
;
toprow: defw    0                ; start of matrix
```

Topsy-Turvy Text
Our character rotation program allows text to be printed vertically, horizontally, and even upside down! The screen shown here gives one example, though the most spectacular effects could be obtained using user-defined graphics

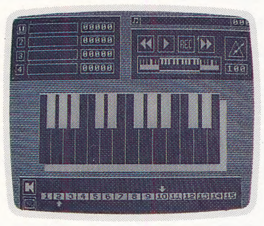


Fast Point Plot

This routine operates in all screen modes and uses the firmware to determine a memory location holding information for the point where the pixel is to be plotted

```
; Entry:  HL contains the physical Y co-ordinate
;         (0-199)
;         DE contains the physical X co-ordinate
;         (0-639)
;         A contains the ink to set the point to
;
; Exit:   All registers preserved
;
dotpos:   equ   #bc1d          ; SCR_DOT_POSITON
encode:   equ   #bc2c          ; SCR_INK_ENCODE
;
plotter:
          push de    ; save the registers
          push hl
          push bc
          push af
;
          call dotpos          ; find the memory byte
          pop  af              ; get the ink back in a
          push af
;
; HL now contains the address of the byte
; and C contains the mask required to set only the
; required pixel on
;
          call encode          ; find the ink mask
;
; The mask to set all pixels to new ink is in a
;
          and  c               ; only want this pixel
;
          ld   b, a            ; save the new ink number
          ld   d, (hl)         ; get the current setting
          ld   a, c            ; get the pixel mask
          cmp  a               ; invert it
          and  d               ; set the pixel to ink 0
;
          or   b               ; mask in the new colour
          ld   (hl), a         ; and store it out
;
          pop  af              ; restore the registers
          pop  bc              ;
          pop  hl
          pop  de
;
          ret
;
```

MAKE YOUR OWN KIND OF MUSIC



Notable Display
The display is just one of the many attractive features of the Music System, using icons and clearly designed menus to help the musician. This shot is taken from the Keyboard Utility program

Most music packages so far produced for micros have required the user to implement complex programs to generate even a single note. The Music System, from Island Software, remedies this problem with a WIMP-based, professional-quality package for the BBC Model B Micro.

The Music System, from Island Logic, contains a suite of five options, each of which can be accessed very easily from disk. Loading from cassette, on the other hand, is complicated by the fact that there are two tapes to deal with. But with its windows, icons and Macintosh-like graphics, the system is both friendly and very attractive.

The first of the five programs, and the one you are most likely to use at first, is the Editor, which the company describes as a 'musician's note-processor'. This makes the process of writing or modifying music exceptionally easy. The screen displays bass and treble staves onto which notes are written; you change the value of the notes, while the bar lines are inserted automatically according to a pre-selected time signature. When you have composed a tune, or just want to hear a few bars played back, you can press the Tab key, which will initiate the audio playback accompanied by a scrolling display on screen.

One of the major attributes of this system is that it is at once suitable for the novice and a great help for the more experienced musician. The beginner can see and hear exactly what is happening at any stage of the composition, and errors are easily spotted, making correction quick and efficient.

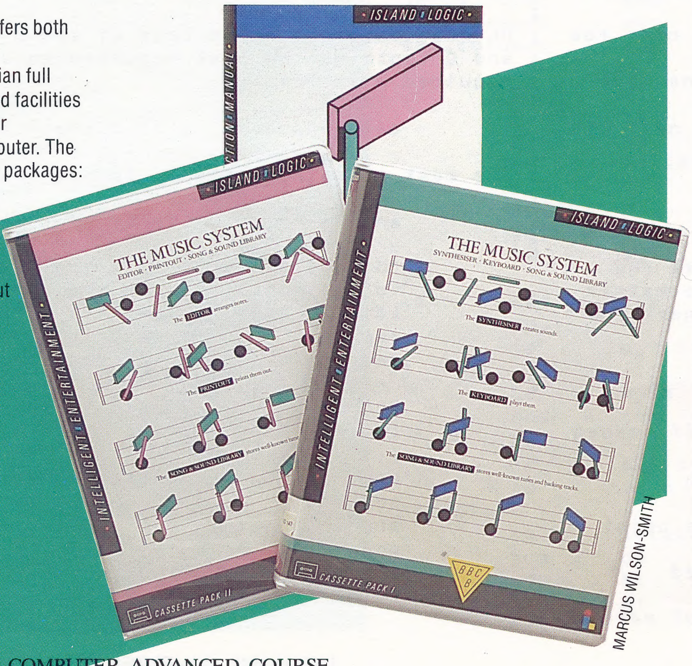
The Song and Sound Library provides a range of backing rhythms to use when composing, as well as a number of well-known tunes that can be played and altered, including Mozart's Piano Sonata in E Flat and Flight of the Bumble Bee. Tempo can be adjusted from the plodding pace of 30 beats per minute, up to the breakneck speed of 200 beats per minute, which could provide some electrifying results if applied to a Bach fugue. If that pace seems a bit extravagant, then you could always put a soft reggae beat behind Mozart's Sonata.

The Synthesiser option, which is used in conjunction with the Editor and Keyboard options, is used to create any sounds you wish. Each sound is defined by setting the parameters of an envelope, with a maximum of 30 envelopes. The disk version also has a choice of 15 different percussive effects. As with the other modules, editing is simple. Each parameter has its own icon on screen, and sounds can be heard as they are entered. Frequency and amplitude graphs can easily be displayed while Synthesiser sounds can be saved to two files and then loaded into the Editor or Keyboard.

The Keyboard option displays the middle two octaves of a full piano keyboard on screen with the QWERTY group on the micro's keyboard acting as the white keys and the numeric keys acting as black. Below the keyboard is a window containing the envelope and volume icons.

One of the most impressive features of this module is its simulation of a four-track tape recorder. Percussion and backing tracks can be laid down and music composed over them, thus making experimentation with your music that much easier.

The Music System is a complex program and includes two thorough manuals that should be studied if you are to get the most out of the package. Once loaded and running, however, the windows and icons instantly demonstrate the ease and efficiency of the system, hopefully leading to greater interest and rewards.



MARCUS WILSON-SMITH

Master Musician
The Music System offers both the experienced and inexperienced musician full control over the sound facilities of the BBC Model B or Commodore 64 computer. The system comes in two packages: the first contains the synthesiser and keyboard modules; the second includes the editor and printout facilities

The Music System: For the BBC(B) and Commodore 64. Cassette Pack 1 – Synthesiser, Keyboard, Song & Sound Library; Cassette Pack 2 – Editor, Printout, Song & Sound Library
Price: £12.95 per pack
Publishers: Island Logic Limited, 22 St Peter's Square, London W6 9NW
Format: Cassette
Joysticks: Not required

FORTH EXTRA

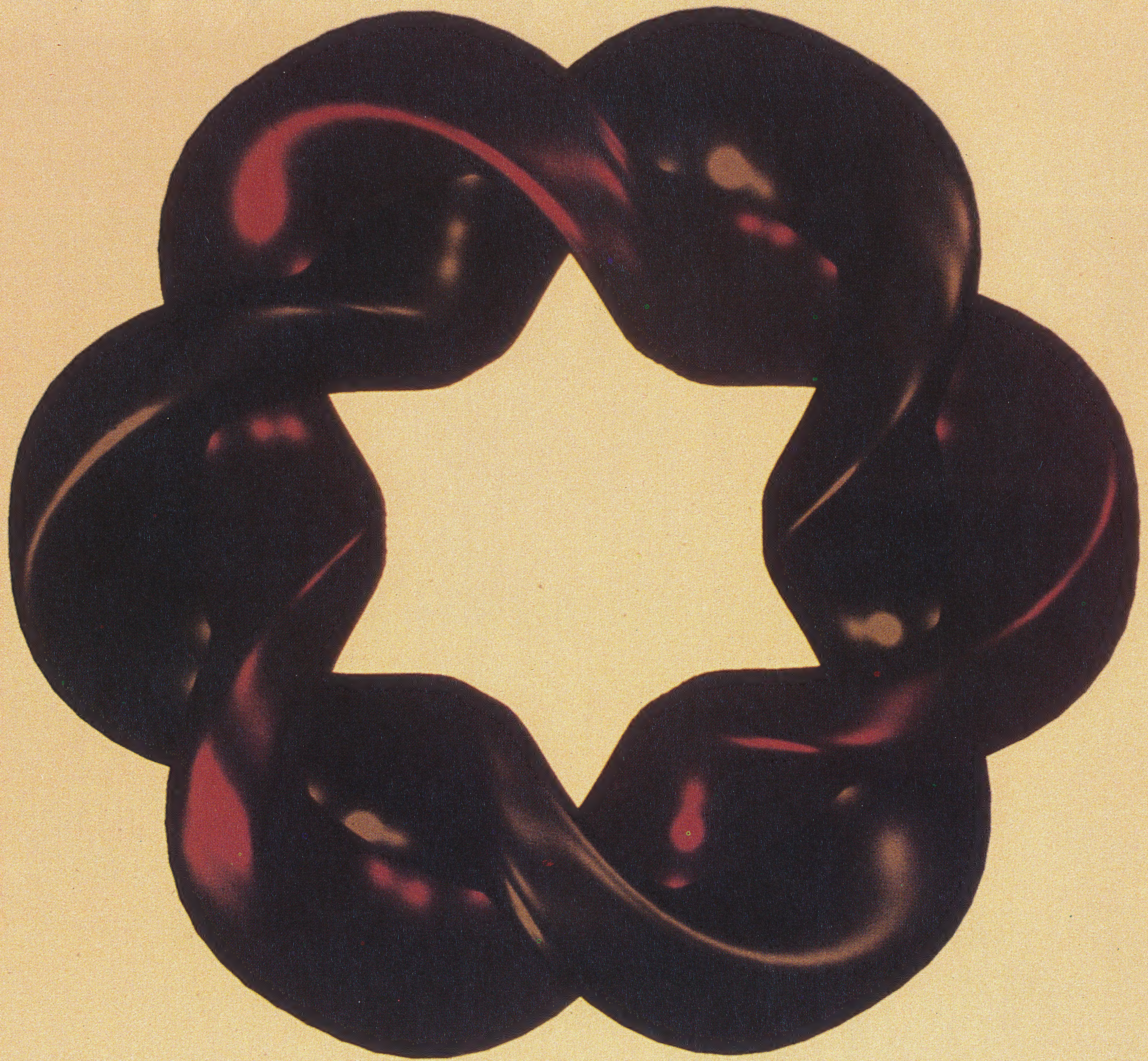
As an extension to our series on FORTH, which was concluded on page 1567, we present a program that fully illustrates the language's most useful feature – its extendability. As we discussed throughout the series, this aspect of FORTH, which accommodates easily manipulated and definable structures, allows it to be used for applications that other languages simply couldn't handle. One such application is the control of machinery, to which our program directly relates.

The machine we are controlling here is a set of traffic lights. If we were doing this for real, it would be through a parallel output port from the computer, controlled by some system-dependent word like OUT (output byte ––). Suppose the three lights are controlled by three bits of this output byte: bit 0 for red, bit 1 for amber and bit 2 for green. If a bit is 1 then its light goes on, if it is 0 then its light goes off. In the example, we replace the proper OUT by one that prints on the screen the names of the lights that are on.

We also need a word that waits for a given time. There is no standard for this, but by experimenting on your computer you can do it using empty DO loops.

```
1 CONSTANT RED
2 CONSTANT AMBER
4 CONSTANT GREEN

: OUT ( lights code –– )
  DUP RED AND IF
    ." red "
  ELSE
    ."   "
  THEN
  DUP AMBER AND IF
    ." amber "
  ELSE
    ."   "
  THEN
  GREEN AND IF
    ." green"
  THEN
  CR
;

VARIABLE LIGHTS ( last lights code written
    OUT )
  0 LIGHTS !
  0 OUT

: ON ( colour code –– )
  LIGHTS @ OR
  DUP OUT
  LIGHTS !
;

OFF ( colour code ––)
  NOT ( use-1 XOR in FORTH-79 or in
    computer ONE FORTH)
  LIGHTS @ AND
  DUP OUT
  LIGHTS !
;

VARIABLE SECLOOPS
30000 SECLOOPS ! ( try varying this )
: 1SEC ( waits one second )
  SECLOOPS @ 0 DO
  LOOP
;

: SECONDS ( seconds to wait ––)
  1 MAX 30 MIN (limit to between 1 and 30
    seconds)
  0 DO
    1SEC
  LOOP
;

: RUN
  RED OFF AMBER OFF GREEN OFF
  5 0 DO
    RED ON 10 SECONDS
    AMBER ON 2 SECONDS
    RED OFF AMBER OFF GREEN ON
      10 SECONDS
    GREEN OFF AMBER ON 2
      SECONDS
      AMBER OFF
    LOOP
  ;
```

Clearly, you now have a very versatile traffic light control language, which you could easily extend to cover an entire junction. It's FORTH's extendability that allows you to mix the traffic light control parts (like RED ON 10 SECONDS) with the program structures in such a readable way.



© 1983 SALESIN, D.—BROWN UNIV.